Drag and Drop: Amalgamation of Authoring, Querying, and Restructuring for Multimedia View Construction

Atsuyuki Morishima†, Seiichi Koizumi††, and Hiroyuki Kitagawa†

†*Institute of Information Sciences and Electronics, University of Tsukuba*
††*Doctoral Program in Engineering, University of Tsukuba*

February 2000

ISE-TR-00-165

Abstract

Recently, it has been a matter of great importance to publish the multimedia data objects stored in various information sources. The World Wide Web is often used as publication media. In such context, a crucial point is how to present the results of the set-at-a-time operation (querying and restructuring of data in underlying information sources). Frameworks to specify how to query and restructure data are usually different from those to specify the presentation of the results in existing systems. This paper proposes a visual user interface which amalgamates authoring, querying, and restructuring functions for multimedia Web view construction. The user is only required to drag and drop data objects, just like in typical authoring tools for HTML and SMIL pages. A feature of our user interface is that the user can designate an existing data object as an example, which will serve as the representative of a set of data objects. Manipulation of an example is interpreted as manipulation of the set of data objects. Therefore, the object-at-a-time authoring framework and the set-at-a-time data manipulation (querying and restructuring) framework are integrated in a seamless way. Another feature is that the interface can cope with semistructured data, which often appear in the context of multimedia view construction. This paper also provides the formal semantics of data operations through the visual user interface.

1. INTRODUCTION

Recently, it has been a matter of great importance to publish the data objects stored in various information sources. The World Wide Web is often used as publication media. The data objects often include not only numerical and text objects, but also multimedia objects such as image, audio, and video objects. Many systems have been proposed in the literature, and a large number of systems are currently used in practice. For example, Web-site management systems such as Strudel (Fernández et al., 1998) can create different Web views on top of heterogeneous information sources. The information sources can be traditional databases and existing Web pages containing multimedia objects. Practical examples are many tools for Web application development. One of their main functions is to show the query result of back-end databases in the form of Web pages.

In such systems, a crucial point is the presentation of the results of the set-at-a-time operation (querying and restructuring of data in underlying information sources). The user is usually required to adopt different schemes to query and restructure data and to present the result. For example, Strudel requires the user to use StruQL for querying and restructuring data, while the HTML template language is used for specifying how to present the result. Also, typical Web application development environments require the user to use SQL for querying and offer visual tools for designing the layout of the result.

This paper proposes a visual user interface which amalgamates authoring, querying, and restructuring functions for multimedia presentation. In our context, construction of multimedia presentation means creation of Web views, involving HTML and SMIL(W3C, 1998)-based multimedia Web documents, on top of heterogeneous information sources. The interface looks like just a common authoring tool for HTML and SMIL documents: The user is only required to drag and drop data objects presented in windows into a blank window named the *canvas*. He can put data objects anywhere he likes, and specify their sizes with mouse operations. As a feature, our interface allows the user to designate an existing data object as an *example*. Then, the data object (the example) serves as the representative of a set of data objects. A drag-and-drop operation of the example is interpreted as manipulation of the set of data objects. Therefore, the object-at-a-time authoring framework and the set-at-a-time data manipulation (querying and restructuring) framework are integrated in a seamless way.

Another important feature of our user interface is that it can cope with *semistructured data* (Abiteboul, 1997)(Buneman, 1997). This feature is important in multimedia integration and presentation. Multimedia objects are often stored and managed in the World Wide Web, which is a well-known example of semistructured data. Because the data structure is often irregular and implicit in semistructured data, the domain of the objects an example represents cannot be fixed in advance. In contrast to this, the domains are fixed in advance in QBE (Zloof, 1977) and other QBE-like query languages

for relational databases. In our framework, the domain is defined dynamically according to the user's interaction with the user interface. By specifying 'another' examples, the user allows the interface to infer the intended domain. This feature is essential for operation of semistructured data.

The visual user interface is originally designed for an information integration system *InfoWeaver*, which we have been developing for heterogeneous information integration (Kitagawa et al., 2000)(Morishima et al., 1999(b)) (Morishima et al., 2000(a)). Although the fundamental design of the interface is independent of InfoWeaver, InfoWeaver provides one of typical contexts where our user interfaces is useful. Figure 1 shows the architecture of InfoWeaver. The mediator (Wiederhold, 1992) and wrappers (Roth et al, 1997) are used for integration. The wrappers provide the mediator with views on top of information sources (based on WebNR/SD, the common data model in our environment). The user manipulates data through the mediator.

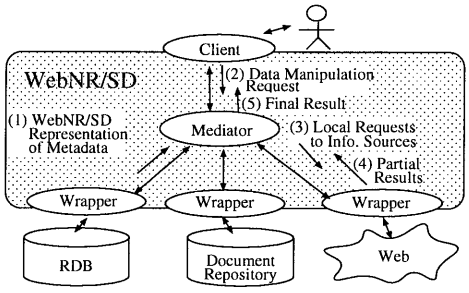

Figure 1 Integration environment InfoWeaver.

The main contributions of this paper are as follows.

1 We propose a visual user interface for constructing multimedia presentation on top of information sources. It amalgamates the object-at-a-time authoring operation and the set-at-a-time querying and restructuring operations.

2 The information sources can be heterogeneous information sources involving semistructured data. Through the interaction with the user, the system infers the target data objects.

3 We present the formal semantics of the data operations through the visual user interface.

The rest of this paper is organized as follows. Section 2 shows an application scenario. Section 3 explains the basic concepts for the visual user interface design. Section 4 shows how to construct the multimedia Web view through the visual user interface. Section 5 explains the formal semantics. Section 6 briefly surveys related work. Section 7 is the conclusion.

2. APPLICATION EXAMPLE

This section shows an example of multimedia Web view construction. The information sources include RDBs and XML-based Web pages. The output is a collection of HTML and SMIL Web pages. First, we show the example scenario. Then, we explain how to use SMIL for multimedia presentation.

2.1. EXAMPLE SCENARIO

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 (RealNetworks, Inc.)) objects and their metadata. The video objects and their metadata are stored in the relation VIDEO(VID, GID, Begin, End, Batter, Pitcher, Contents). VID is a Video ID. The domain of the attribute Contents is an ADT for RealMedia, named VIDEO type. The other attributes GID, Begin, End, Batter, and Pitcher represent metadata on the Contents. The meaning of the relation VIDEO is that each VIDEO value in Contents records a scene starting from the Begin time to the End time of a game (represented by GID), where Batter and Pitcher are facing each other. (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(P-Name, 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(a). 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. We also assume that the pages on the site are written in XML and that there are two variations in the structure of the team pages. Figure 3 shows the two variations in page structure. One has flat structure, while the other groups players into categories.

The requirement here is to create a multimedia Web view on top of the above information sources. A SMIL Web page is constructed for each player whose batting average is more than 0.3 for the current season. It is a *multimedia page* (Figure 2(b)), which consists of three different kinds of components: (1) *A sequential rendering of scenes (video objects)* in which he is at bat. (2) *The logo of the team he belongs to.* (3) *Text description of his profile.*

An index HTML page is also created (Figure 2(c)). It contains an image object ('GoodBatters'), the selected players' names, batting averages, and links to the players' multimedia pages.

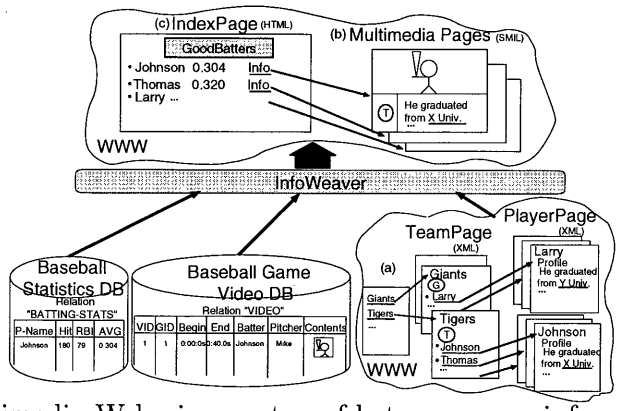

Figure 2 Multimedia Web view on top of heterogeneous information sources.

```
<team>
  <tname>Tigers</tname>
  <logo><img src="TigersLogo.gif"/></logo>
  <players>
    <player ppage="http://..">Johnson</player>
    <player ppage="http://..">Thomas</player>
    ..
  </players>
<team>
```

(a) Team page of Tigers

```
<team>
  <tname>Giants</tname>
  <logo><img src="GiantsLogo.gif"/></logo>
  <players>
    <fielders>
      <player ppage="http://..">Larry</player>
      ..
    </fielders>
    <pitchers>
      <player ppage="http://..">Brian</player>
      ..
    </pitchers>
  </players>
</team>
```

(b)Team page of Giants

Figure 3 Team page variations.

2.2. SMIL

SMIL (Synchronized Multimedia Integration Language) can represent integrated multimedia contents as XML-based tagged text. Figure 4 is an example of a SMIL page, that represents a multimedia page explained in Subsection 2.1. A SMIL page must start with the tag `<smil>` and end with `</smil>`. In general, we refer to a substring which is surrounded by <*g*> and </*g*> tags as *an element*. The <*g*> tag of an element often contains attributes of the element.

```
<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="Scene_J1.rm" region="R1"/>
        <video src="Scene_J2.rm" region="R1"/>
      </seq>
      <img src="TigersLogo.gif" region="R2"/>
      <textstream src="ProfileJ.rt" region="R3"/>
    </par>
  </body>
</smil>
```

Figure 4 Example of a SMIL page.

The page has two main parts. First, the head part specifies the layout of the multimedia page. In this example, the layout part says the page contains three rectangular regions. The attributes used in the head part are as follows. 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, respectively. These tags have the following attributes. The `src` attribute represents the URL of the referenced data object. The `region` attribute represents the region where the data object is rendered.

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 example SMIL page specifies that the multimedia presentation shows in parallel a sequence of two scenes (video objects), an image object, and a textstream object.

3. BASIC CONCEPTS

This section explains the basic concepts involved in the visual user interface design. Formal definitions are given in Section 5.

3.1. WINDOWS

The interface consists of three types of windows.

DataBox: A *DataBox* is used to display a set of data items stored in an information source. Figure 5 shows example DataBoxes. The DataBoxes (a) and (b) are used to display relations in relational databases. In this case, each DataBox is connected to a relation, and the display unit is a tuple. The user can click the Next and Previous buttons to browse other tuples in the relation. The relation to be displayed is designated by using the Open menu of the DataBox. The DataBoxes (c) and (d) are used to display Web pages. In this case, each DataBox is associated with one or more Web pages. The display unit is a Web page. The Web page(s) to be displayed is designated either by specifying a URL or by using a mechanism to gather Web pages. This paper assumes that we can gather Web pages using some mechanism. We reported a page gathering mechanism via browsing and querying in (Morishima et al., 1999(a)). Also, Web query languages such as WebSQL (Mendelzon et al., 1996) can be used for this purpose. But we omit the details because it is beyond the scope of this paper. A DataBox displays a unit of data items (a tuple or a Web page) at a time. In the remaining part of this paper, we call a display unit in a DataBox just *a page.*

Palette: A *Palette* is a window which provides various component data objects to be included in the data manipulation result. Figure 6 is an example of a Palette. The palette contains an image object, a hypertext link object, a dot object to represent a list item, a horizontal rule object, and so on.

Canvas: The *Canvas* is a blank window into which the user can drag-and-drop data objects from DataBoxes and Palettes. The user can put data objects anywhere he likes, and specify their sizes with mouse operations.

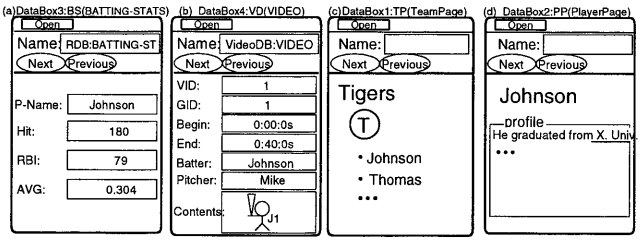

Figure 5 DataBoxes.

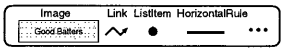

Figure 6 Palette.

3.2. OBJECTS

In our context, an *object* is the unit of drag-and-drop operation. For example, an element (a substring which is surrounded by <*g*> and </*g*> tags) contained in a Web page and an attribute value in a relation are objects.

3.3. DRAG-AND-DROP

Drag-and-drop is the basic operation of this interface. By dragging-and-dropping objects from DataBoxes and Palettes into the Canvas, the user can construct various multimedia Web views on top of heterogeneous information sources. (See Figure 7.) In Figures 7~12, multiple pages are shown simultaneously in a DataBox for explanatory purposes. Actually, the user has to press Next and Previous buttons to see them.

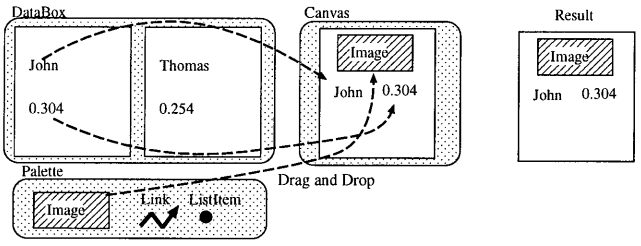

Figure 7 Drag-and-Drop operation and the result.

3.4. EXAMPLES

Introduction of the concept of examples into a drag-and-drop-based page authoring framework is the distinguishing feature of our user interface design. The user can designate an object as an example by clicking a mouse button on the object. (We explain this in Section 4.) We call the object specified as an example an *example object*, or shortly, an *example*. A drag-and-drop operation of an example is interpreted as manipulation of a set of objects the example represents. Therefore, the object-at-a-time authoring framework and the set-at-a-time data manipulation (querying and restructuring) framework are integrated in a seamless way.

3.5. TARGET SETS

A set of objects an example represents is called the *target set* of the example. As a default, the target set is defined as the set of objects each of which appears at the same position on a page as the example object. Manipulation of an example means manipulation of objects in its target set (Figure 8).

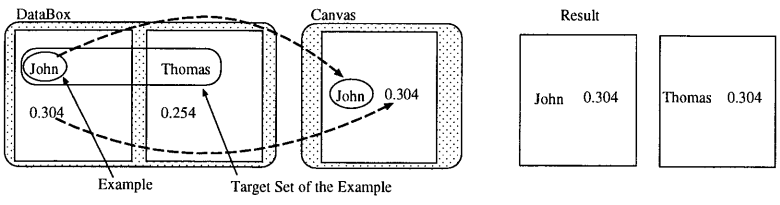

Figure 8 Manipulation of an example and the result.

3.6. ASSOCIATIONS

When the user specifies multiple examples (and their target sets), it is often the case that *associations* occur among the target sets. Two types of associations are considered in this interface design. The first one is the *structural association* (S-Association). This occurs according to a structural relationship (relative position) between two examples. For example, if two examples are on the same page, it implies an S-Association so that objects taken from their target sets must reside on the same page. (Actually, S-Association can have more general meaning. We explain this in Section 5.) The second one is the *value association* (V-Association). This occurs if values of example objects are the same. If any association occurs among the target sets, only some combinations of objects are qualified to be manipulated. Note that an association serves as a kind of join condition.

For example, suppose that the user wants to change the page layout in the DataBox in Figure 8 so that the name and the batting average appear side by side. If the user takes example objects as in Figure 9, the system considers that there is no S-Association between the two target sets. Therefore, all combinations of the objects appear in the result. In contrast, if he takes examples as in Figure 10, S-Association occurs and he gets the intended result.

Then, suppose that we have two DataBoxes as shown in Figure 11, and that the user wants the pages each of which contains a player's name, his team name, and his batting average. If he takes example objects as in Figure 11, there is no V-Association. However, if he takes examples as in Figure 12, V-Association occurs and he gets the intended result.

We explain the formal semantics in Section 5.

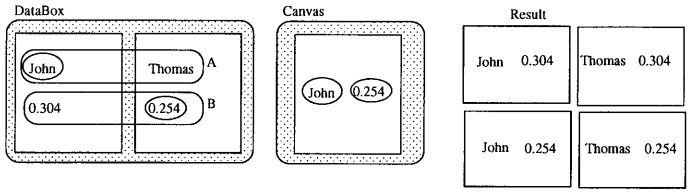

Figure 9 No S-Association between target sets A and B.

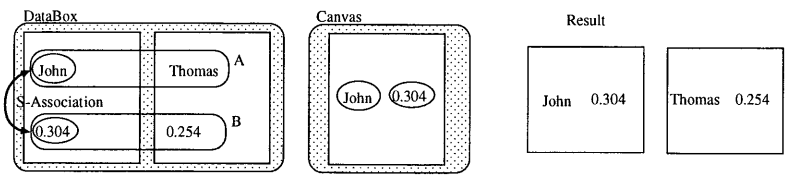

Figure 10 S-Association between two target sets A and B.

(Only the pair of the objects on the same page are considered.)

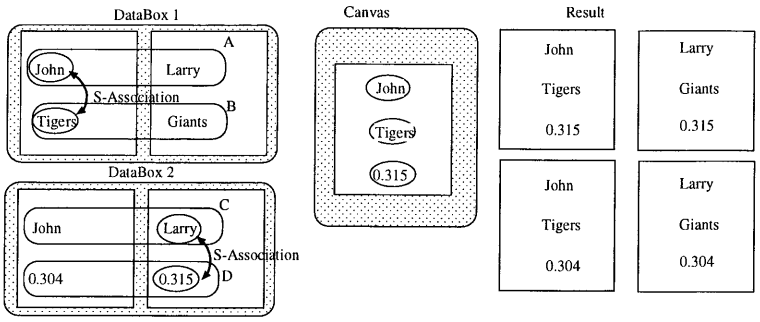

Figure 11 No V-Association.

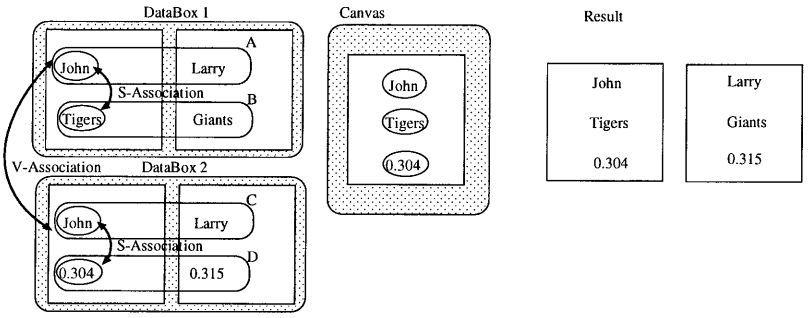

Figure 12 V-Association between target sets A and C.
(Only objects with the same value match each other.)

4. OPERATION

This section explains how to manipulate data through this user interface, and gives specifications for the example scenario shown in Section 2.

4.1. EXAMPLES AND TARGET SETS

Basically, what the user has to do is just to drag-and-drop objects from DataBoxes and Palettes into the Canvas, and arrange the objects as he likes. However, as mentioned before, the user can specify that an object is an *example* before dragging-and-dropping the object. If the user clicks the right mouse button on an object in a DataBox, a menu appears (Figure 13). Selecting the "Example" menu item specifies that the object is an example. At first, the system assigns a *default target set* to the example object. It is the set of objects each of which resides at the same position on a page as the example. For example, in Figure 13, the target set would be the set of the P-Name values in Relation BATTING-STATS. The "Another" and "Clue" menu items are used to modify the target set.

Example objects are always highlighted in DataBoxes. If the user clicks the left mouse button on an example object, objects in its target set are also highlighted. Therefore, the user can identify non-example objects, example objects, and their target sets anytime.

The following regular expression shows the operation procedure through our user interface.

('Example' ('Another' | 'Clue')* | 'D&D')*

Here, 'Example,' 'Another,' and 'Clue' mean selections of respective menu items on an object. The 'D&D' means Drag-and-Drop operation. We call them 'Example,' 'Another,' 'Clue,' and 'D&D' operations, respectively. Intuitively, the user interface allows any combinations of the following operation patterns.

- To specify that an object is an example, and accept the target set.
- To specify that an object is an example, and change the default target set by successive 'Another' and 'Clue' operations.
- To drag and drop an non-example object (an object which the user has not specified as an example or one in a Palette), and arrange it on the Canvas.
- To drag and drop an example object, and arrange it on the Canvas.

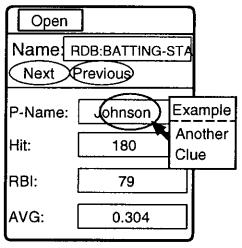

Figure 13 Menu to specify examples and target sets.

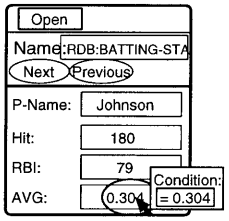

Figure 14 ClueBox.

If 'Another' operation is performed after 'Example' operation, the target set of the example is extended to include the 'Another' object. Intuitively, the system tries to generalize the relationship between the position of the example and that of the 'Another' object, and includes objects which have the generalized relationship with the example into the target set. (We show the rules in Subsection 5.2.)

If 'Clue' operation is performed after 'Example' operation, the system narrows the target set. First, the *ClueBox* appears on the display (Figure 14). The user can change the shown condition into another one. For example, he can change the condition "$= 0.304$" into "> 0.3." In general, 'Clue' operation makes the target set contain only the objects o_i each of which satisfies all the following conditions.

- o_i is included in the original target set.
- o_i has a corresponding object c_i such that they have the same relative position as the example object and the 'Clue' object have. (In the above example, c_i is the batting average of each player.)
- c_i satisfies the condition specified in the ClueBox. (In the example, the batting average value has to be more than 0.3)

Therefore, in the example, the target set is narrowed to contain only the players whose batting averages are more than 0.3.

4.2. ASSOCIATIONS

As mentioned in Subsection 3.6, S-Association occurs according to the structural relationship (relative position) between two example objects. V-Association occurs if two example objects have the same value.

4.3. GROUPING

By default, one result page is generated for each (qualified) combination of objects (See Figures 9~12). This rule can be changed by putting the repetition mark (*) at the appropriate position on the Canvas. Essentially, it works as Nest operator of the nested relational algebra (Fischer, 1983). The following subsection includes examples of grouping.

4.4. OPERATIONS FOR THE EXAMPLE SCENARIO

Figure 15 illustrates specification to obtain the required result of the example scenario given in Section 2. We assume here that DataBoxes TP, PP, BS, and VD contain all the team pages, all the player pages, Relation BATTING-STATS, and Relation VIDEO, respectively. We show the operation sequence.

(1) Open the Canvas and declare the construction of an HTML page. (Then, the system opens a space for the HTML page on the Canvas.)

(2) D&D the image 'Good Batters' from the Palette into the Canvas.

(3) D&D the 'ListItem' object from the Palette into the Canvas.

(4) Specify that 'Johnson' in TP is an example. The default target set includes those players who appear first in the player list of each team page having the structure shown in Figure 3(a). Next, specify that 'Thomas' in TP is an 'Another' object. Then, press the Next button of the DataBox TP to find the team page of Giants, and specify that 'Larry' on the Giants' page is an 'Another' object. (Alternatively, you can use another page which has the structure shown Figure 3(b).) The system uses rules to generalize the relationship between positions of 'Johnson,' 'Thomas,' and 'Larry,' so that the target set of this example is extended to include all players of all teams.

(5) D&D the "Johnson" from TP into the Canvas.

(6) Put a repetition mark (*) on the list item object. As a result, all the players are listed in this page. Otherwise, a new page is produced for each player.

(7) Specify that 'Johnson' objects in PP and VD are examples. Note that the three target sets of 'Johnson' objects in TP, PP, and VD have V-Association. Therefore, this specifies equality joins between their target sets.

(8) Specify that 'Johnson' in BS is an example. (Its target set also has V-Association with the above three target sets.) Then, specify that '0.304' in BS is a clue of the example. Rewrite the condition in the ClueBox and make it '> 0.3' so that the target set of the example 'Johnson' in BS includes only players with their batting averages over 0.3.

(9) Specify that '0.304' in BS is an example. D&D it from BS into the Canvas.

(10) Declare the construction of a SMIL page. The system opens a space for the SMIL page on the Canvas.

(11) D&D the HypertextLink object from the Palette into the Canvas. Connect it to the SMIL page.

(12) Specify that '  ' in VD is an example. D&D it into the Canvas.

(13) Put the repetition mark (*) on the dropped '  ' object. As a result, the scenes (video objects) of a player are rendered sequentially as one video. Otherwise, a SMIL page is produced for each scene.

(14) Specify that '  ' in TP is an example. D&D it into the Canvas.

(15) Specify that the profile in PP is an example. D&D it into the Canvas.

(16) Press the 'Create' button on the Canvas.

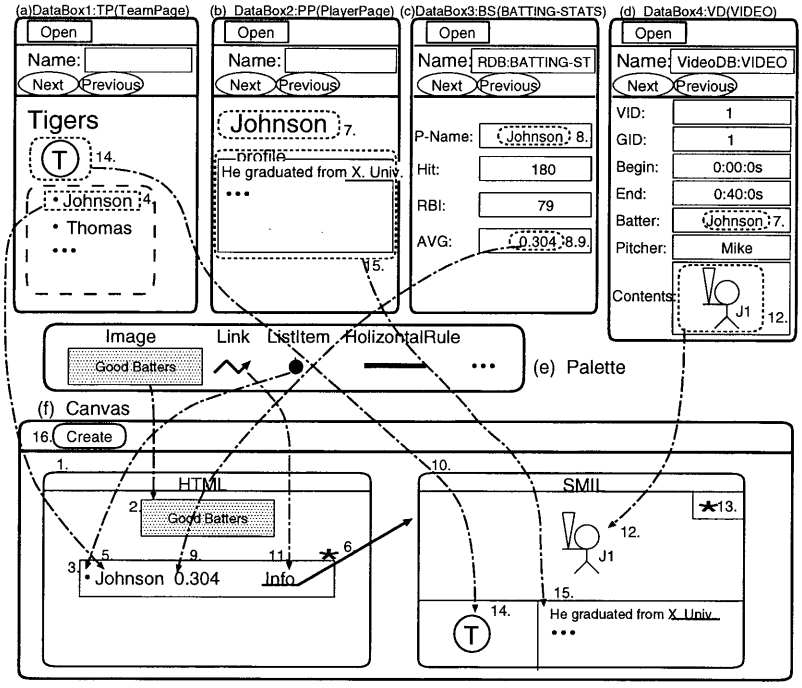

Figure 15 Specification for the example scenario.

Figure 16 is a screen shot of the prototype system under development where the user is doing the example scenario operations.

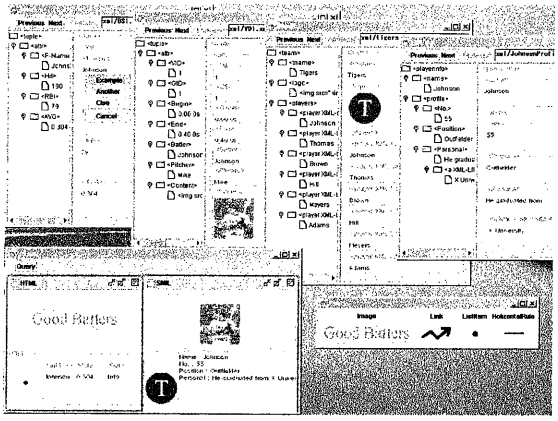

Figure 16 A screen shot from the prototype.

5. SEMANTICS

This section gives the formal semantics for the user interface in terms of the predicate logic and the nested relational algebra (Fischer, 1983). The formal semantics defines the behavior of the user interface more precisely. Therefore, it is not only of use for theoretical analyses, but it also makes the proposed scheme to be more easily applicable to various contexts, such as semistructured databases and CGI-based Web page generators.

We define the formal semantics in the following three steps.

1 We express the source data as an object tree.

2 We derive the *target relation* according to the user's interaction. The target relation represents target sets and the associations among objects in the target sets.

3 We restructure the relation into a nested relation which reflects the grouping structure specified by repetition marks on the Canvas. This nested relation specifies the final result.

5.1. DATA MODELING

We represent the source data as an object tree. The tree in Figure 17 represents a part of the source data in the example scenario shown in Section 2.

Every node is annotated with a *label*, which consists of a *label name* and a *label number*. (We omit the root label for simplicity.) The followings are some remarks. (1) Subtrees whose roots are second level nodes (children of the root) correspond to DataBoxes. They have labels in the form of 'DB.1,' where DB is the name of the corresponding DataBox. Subtrees whose roots are third level nodes correspond to pages. They are labeled with 'PAGE.i.' (2) A subtree whose root is a fourth level node represents an XML page or a tuple of a relation. (3) The label number of a node is 1 if none of its sibling nodes has the same label name. Otherwise, label numbers are sequentially assigned to sibling nodes with the same label name. (4) We refer to the nodes of the tree as *objects* and tag them with OIDs. In Figure 17, some OIDs are

explicitly presented in the form of &n for the convenience of the following explanation. (5) Note that this is *semistructured data*. There are two kinds of team pages whose structure is different. A PLAYER may be a direct child of a PLAYERS element or be placed under other elements such as FIELDERS and PITCHERS.

In the following discussion, $path(o)$ and $value(o)$ denote the path from the root to the object o and the value of o, respectively. For example, $path(\&12)$ = TP.1→PAGE.1→TEAM.1 →PLAYERS.1→PLAYER.1, and $value(\&12)$ = "Johnson." We often reference an object by its value if there is no ambiguity.

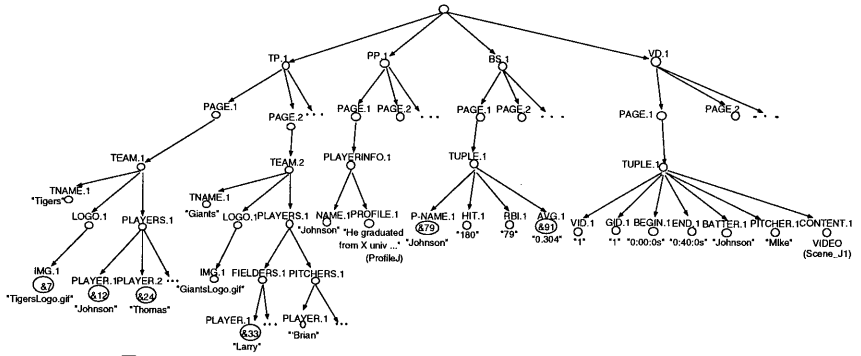

Figure 17 Tree representation of the source data.

5.2. TARGET SETS

Each example object has a corresponding target set. Given an example e, its *target set* (denoted by TS_e) is defined as follows.

Case 1: If no 'Clue' operation has been invoked for e,

$$TS_e = \{o | o \in O \wedge Candidate\text{-}Pred_e(o)\},$$

where O is the set of all the objects in the object tree, and $Candidate\text{-}Pred_e(o)$ is a *candidate predicate* incorporating a path expression. A path expression is similar to a path but may contain wildcards. $Candidate\text{-}Pred_e(o)$ holds if and only if $path(o)$ conforms to the path expression. The candidate predicate is determined by the 'Example' and 'Another' operations as shown below.

Example The following $TS_{\&12}$ gives the target set specified by Operation (4) in Subsection 4.4.

$TS_{\&12} = \{o | o \in O \wedge$TP.1→PAGE.?→TEAM.1→PLAYERS.1→?*→PLAYER.?$[o]\}$

Wildcards considered in the paper are listed in Figure 18. Given the source data shown in Figure 17, $TS_{\&12}$ = {"Johnson", "Thomas", ..., "Larry", ..., "Brian", ...} (all players of all teams). Note that the wildcard '?' matches with any label number, and that '?*' with any sequence of any nodes ('?*' also matches with the sequence whose length is 0). Therefore, all players in the two different kinds of team pages are included.

Wildcard	What to match with
Name.?	A node with label name *Name*
?	A node with any label
?*	Any sequence of any nodes (the length can be 0)

Figure 18 Wildcards.

Derivation of Candidate Predicates *Candidate-Pred*$_e(o)$ is determined by 'Example' and 'Another' operations. In the derivation process, paths and values of objects play important roles. For this purpose, we add *annotations* to predicates. Annotations for *Candidate-Pred*$_e(o)$ give information on *path*(e) and *value*(e)[2]. Annotations are surrounded by "⟨" and "⟩"[3]. We represent the null sequence as ε.

For example, specification of $TS_{\&12}$ with annotations is as follows.

$$TS_{\&12} = \{o | o \in O \wedge \text{TP.1}\rightarrow\text{PAGE.?}\langle\text{PAGE.1}\rangle \rightarrow\text{TEAM.1}\rightarrow\text{PLAYERS.1} \rightarrow?^*\langle\varepsilon\rangle \rightarrow\text{PLAYER.?}\langle\text{PLAYER.1}\rangle[o\langle\text{Johnson}\rangle]\}$$

Note that the annotations give information on *path*(&12) and *value*(&12).

In general, *Candidate-Pred*$_e(o)$ is derived as follows.

(1) First, when the user specify that the object e is an example, the *default candidate predicate* $p[o\langle v\rangle]$ is derived. Here, p is same as *path*(e) except that its PAGE.i is replaced by PAGE.?⟨PAGE.i⟩, and v is *value*(e).

For example, consider Operation (4) in Subsection 4.4. When the user specifies the object &12 (with its value "Johnson") as an example, the default candidate predicate derived is TP.1→PAGE.?⟨PAGE.1⟩ → TEAM.1 → PLAYERS.1→ PLAYER.1$[o\langle Johnson\rangle]$. The predicate defines the default target set explained in Subsection 4.1. That is, the set of objects which appear at the same position on different pages.

(2) 'Another' operations modify the candidate predicate to accept the 'Another' objects. As mentioned in Section 4, the system tries to infer the target set the user intends according to the example and 'Another' objects. For this purpose, we use modification rules as shown in Figure 19. Each rule prescribes how to modify the original predicate according to *path*(e) and *path*(a), where e and a are the example and an 'Another' object, respectively. Note that we can get *path*(e) from the original predicate because it has annotations. In Figure 19, B and C denote label names, q_i denotes a partial path, and p_i denotes the partial path expression of the original predicate which q_i conforms to. q_i' is a partial path that p_i can accept. The basic idea behind the rules is to place a wildcard at the position where *path*(e) and *path*(a) conflict with each other.

For example, in Operation (4), the user specifies the object &24 (with its value "Thomas") as the first 'Another' object. Then, *path*(a) = TP.1→PAGE.1 →TEAM.1 →PLAYERS.1→PLAYER.2. We can obtain *path*(e) = TP.1→ PAGE.1→TEAM.1→ PLAYERS.1→PLAYER.1 from the annotated default candidate predicate. The system finds that the default candidate predicate cannot accept *path*(a) because PLAYER.1 in the path expression conflicts with PLAYER.2. In this case, they conflict at their label numbers. Therefore, we can apply Rule 1. Here, p_1 = TP.1→PAGE.?⟨PAGE.1⟩→ TEAM.1 →PLAYERS.1, $q_1 = q_1'$ = TP.1→PAGE.1→TEAM.1 →PLAYERS.1, and $p_2 = q_2 = q_2' = \varepsilon$. The modified predicate becomes TP.1→PAGE.?⟨PAGE.1⟩→ TEAM.1→ PLAYERS.1→PLAYER.?⟨PLAYER.1⟩$[o\langle$Johnson$\rangle]$. Next, the user specifies the object &33 (with its value "Larry") as the second 'Another' ob-

ject. In the similar way, we can apply Rule 3 to the modified predicate. This results in the above $TS_{\&12}$.

	Original Pred.	$Path(e)$	$Path(a)$	Modified Pred.
Rule 1	$p_1\ B.i\ p_2[o\langle v\rangle]$	$q_1\ B.i\ q_2$	$q_1'\ B.k(\neq i)\ q_2'$	$p_1\ B.?\langle B.i\rangle\ p_2[o\langle v\rangle]$
Rule 2	$p_1\ B.i\ p_2[o\langle v\rangle]$	$q_1\ B.i\ q_2$	$q_1'\ C(\neq B).k\ q_2'$	$p_1\ ?\langle B.i\rangle\ p_2[o\langle v\rangle]$
Rule 3	$p_1\ q_3\ p_2[o\langle v\rangle]$	$q_1\ q_3\ q_2$	$q_1'\ q_4(\neq q_3)\ q_2'$	$p_1? * \langle q_3\rangle\ p_2[o\langle v\rangle]$

Figure 19 Rules for modification of candidate predicates. (p_i, q_i and q_i' can be ε.)

Case 2: If a 'Clue' object cl is specified for e with 'Clue' operation,

$$TS_e = \{o | o \in O \wedge Candidate\text{-}Pred_e(o) \wedge \exists c \in O(Clue\text{-}Pred_{cl}(c) \wedge SA\text{-}Pred_{e,cl}(o,c))\},$$

where O is the set of all the objects in the object tree, $Candidate\text{-}Pred_e(o)$ is the candidate predicate which is derived according to the 'Example' and 'Another' operations as explained above. The *clue predicate* $Clue\text{-}Pred_{cl}(c)$ and *S-Association Predicate* $SA\text{-}Pred_{e,cl}(o,c)$ are derived from the 'Clue' operation. $Clue\text{-}Pred_{cl}(c)$ is also a predicate incorporating a path expression with wildcards. The difference is that it prescribes a condition on $value(c)$. The predicate $SA\text{-}Pred_{e,cl}(o,c)$ constrains the relative position of o and c. Intuitively, an object o in the target set has to satisfy the candidate predicate, and be accompanied by an object c which satisfies the following conditions as well.

- o and c have the same structural association (relative position) as the example object e and the clue object cl.
- c satisfies the condition specified in the ClueBox.

Example Operation (8) for the example scenario derives the following target set. (It contains annotations, too).

$TS_{\&79} = \{o | o \in O \wedge p\rightarrow\text{P-NAME.1}[o\langle\text{Johnson}\rangle] \wedge \exists c \in O(p\rightarrow\text{AVG.1}[c > 0.3] \wedge Share_p(o,c))\}$

where p = BS.1→PAGE.?⟨PAGE.1⟩→TUPLE.1.

In the above example, "BS.1→PAGE.?⟨PAGE.1⟩→TUPLE.1→AVG.1[$c >$ 0.3]" is the clue predicate. The predicate holds if and only if $path(c)$ conforms to the path expression and $value(c)$ is more than 0.3. $SA\text{-}Pred_{e,cl}(o,c)$ has the form $Share_p(o,c)$, where p is a path expression. $SA\text{-}Pred_{e,cl}(o,c)$ in the above example is $Share_{BS.1\rightarrow PAGE.?\rightarrow TUPLE.1}(o,c)$. It holds if and only if $path(o)$ and $path(c)$ share the same partial path which starts from the root and conforms to the path expression "BS.1→ PAGE.?→TUPLE.1" [4]. (See Figure 20.) Therefore, $TS_{\&79}$ contains only the players whose batting average is more than 0.3. Like candidate predicates, clue predicates can be tagged with annotations to record $path(cl)$. Thus, we can obtain $path(e)$ and $path(cl)$ from annotated candidate and clue predicates. In the above example, $path(e)$ = $path$(&79 (with the value 'Johnson'))=BS.1→PAGE.1→ TUPLE.1→P-NAME.1, and $path(cl)$ = $path$(&91 (with the value '0.304'))= BS.1→ PAGE.1→ TUPLE.1→AVG.1.

Detailed description of the predicate derivation in the case 2 is given in the appendix.

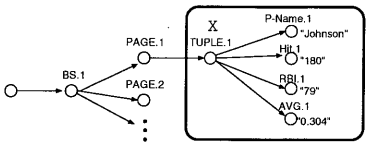

Figure 20 An object set X such that $(\forall o_i \forall o_j \in X) Share_{BS.1 \rightarrow PAGE.? \rightarrow TUPLE.1}(o_i, o_j)$.

5.3. TARGET RELATION

A target relation represents the target sets and the associations among them.

Definition Assume that there are n target sets without clues (specified by examples $e_1, \ldots e_n$), m target sets with clues (specified by examples $e_{n+1}, \ldots, e_{n+m}$ and clues $c_1, \ldots, c_m$), and l associations among them. Then, the target relation is defined as follows.

$$
\begin{aligned}
TR = \{ & (value(o_1), \ldots, value(o_{n+m})) \mid \\
& o_1 \in O \wedge Candidate\text{-}Pred_{e_1}(o_1) \\
& \wedge \ldots \\
& \wedge o_n \in O \wedge Candidate\text{-}Pred_{e_n}(o_n) \\
& \wedge o_{n+1} \in O \wedge Candidate\text{-}Pred_{e_{n+1}}(o_{n+1}) \wedge \exists c_1 \in O(Clue\text{-}Pred_{cl_1}(c_1) \\
& \qquad \wedge SA\text{-}Pred_{e_{n+1}, cl_1}(o_{n+1}, c_1)) \\
& \wedge \ldots \\
& \wedge o_{n+m} \in O \wedge Candidate\text{-}Pred_{e_{n+m}}(o_{n+m}) \wedge \exists c_m \in O(Clue\text{-}Pred_{cl_m}(c_m) \\
& \qquad \wedge SA\text{-}Pred_{e_{n+m}, cl_m}(o_{n+m}, c_m)) \\
& \wedge Association\text{-}Pred_1(o_{a_1}, o_{b_1}) \wedge \ldots \wedge Association\text{-}Pred_l(o_{a_l}, o_{b_l}))\},
\end{aligned}
$$

where $Association\text{-}Pred_i$ is an S-Association predicate or V-Association predicate. As explained before, S-Association predicate has the form $Share_p(o, o')$. V-Association predicate has the form $o \stackrel{v}{=} o'$, and holds if and only if $value(o) = value(o')$. S-Association predicates are determined by paths of example objects and candidate predicates. V-Association predicates are derived from the values of example objects.

Example The target relation for the example scenario is shown in Expression A. (It shows annotations, too.) The superscript number indicates the operation number in Subsection 4.4 to which the predicate corresponds.

$$
\begin{aligned}
& \{(value(o_1), \\
& \ldots, value(o_8)) \mid o_1 \in O \wedge p_1 \rightarrow \text{PLAYERS.1} \rightarrow ?^* \langle \varepsilon \rangle \rightarrow \text{PLAYER.?} \langle \text{PLAYER.1} \rangle [o_1 \langle Johnson \rangle]^{(4)} \\
& \qquad \wedge o_2 \in O \wedge p_1 \rightarrow \text{LOGO.1} \rightarrow \text{IMG.1}[o_2 \langle \textcircled{T} \rangle]^{(14)} \\
& \qquad \wedge o_3 \in O \wedge p_2 \rightarrow \text{NAME.1}[o_3 \langle Johnson \rangle]^{(7)} \\
& \qquad \wedge o_4 \in O \wedge p_2 \rightarrow \text{PROFILE.1}[o_4 \langle ProfileJ \rangle]^{(15)} \\
& \qquad \wedge o_5 \in O \wedge p_3 \rightarrow \text{P-NAME.1}[o_5 \langle Johnson \rangle]^{(8)} \\
& \qquad \qquad \wedge \exists c_1 \in O(p_3 \rightarrow \text{AVG.1}[c_1 > 0.3]^{(8)} \wedge Share_{p_3}(o_5, c_1)^{(8)}) \\
& \qquad \wedge o_6 \in O \wedge p_3 \rightarrow \text{AVG.1}[o_6 \langle 0.304 \rangle]^{(9)} \\
& \qquad \wedge o_7 \in O \wedge p_4 \rightarrow \text{BATTER.1}[o_7 \langle Johnson \rangle]^{(7)} \\
& \qquad \wedge o_8 \in O \wedge p_4 \rightarrow \text{CONTENT.1}[o_8 \langle \boxed{\text{오}\atop\text{지}} \rangle]^{(12)} \\
& \qquad \wedge Share_{p_1}(o_1, o_2)^{(14)} \wedge Share_{p_2}(o_3, o_4)^{(15)} \wedge Share_{p_3}(o_5, o_6)^{(9)} \\
& \qquad \wedge Share_{p_4}(o_7, o_8)^{(12)} \wedge o_1 \stackrel{v}{=} o_3^{(7)} \wedge o_1 \stackrel{v}{=} o_7^{(7)} \wedge o_1 \stackrel{v}{=} o_5^{(8)}\}
\end{aligned}
$$

where

$p_1 =$ TP.1→PAGE.?⟨PAGE.1⟩→TEAM.1,
$p_2 =$ PP.1→PAGE.?⟨PAGE.1⟩→PLAYERINFO.1,
$p_3 =$ BS.1→PAGE.?⟨PAGE.1⟩→TUPLE.1, and
$p_4 =$ VD.1→PAGE.?⟨PAGE.1⟩→TUPLE.1.

Expression A. Specification of the target relation with annotations.

Figure 21 shows all the target sets and associations involved in the example scenario. Figure 22 shows the target relation based on the target sets and associations.

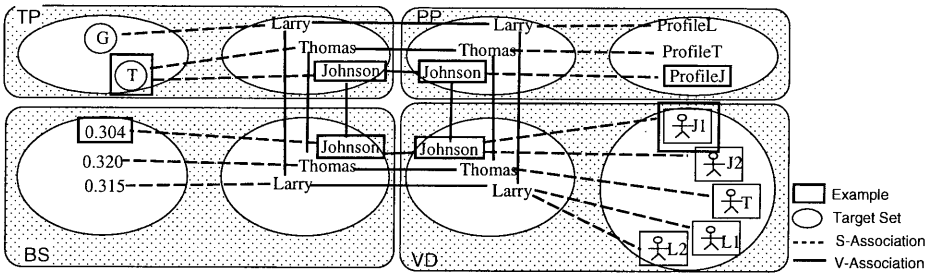

Figure 21 Target sets and associations.

Derivation of Association Predicates Association predicates among target sets are derived when candidate predicates are created or modified. We explain the derivation rules below.

(1) S-Association Predicate: Let e_i and e_j be example objects. The system searches for the longest path expression p which satisfy the following conditions.

- p is a common prefix path expression of $Candidate\text{-}Pred_{e_i}(o_i)$ and $Candidate\text{-}Pred_{e_j}(o_j)$
- $path(e_i)$ and $path(e_j)$ share the same prefix partial path in the range of p.

If it can find p, it derives $Share_p(o_i, o_j)$ as an S-Association Predicate. The principle behind this rule is that if two example objects e_i and e_j share the same prefix path, and the corresponding path expressions (including wild cards) are the same (this is p), then the system infers that the user wants to associate an object o_1 in one target set with o_2 in the other target set when they share the partial path in the same way as e_i and e_j.

For example, look at Expression A and consider two examples e_1 ('Johnson' in TP, &12) and e_2 (' Ⓣ ' in TP, &7). In this case, $path(e_1)$ =TP.1→PAGE.1 →TEAM.1→ PLAYERS.1 →PLAYER.1, and $path(e_2)$ =TP.1→PAGE.1→ TEAM.1→LOGO.1→ IMG.1. Therefore, the system finds that the 'Johnson' and 'Ⓣ' appear on the same page. (More precisely, the example objects share the prefix partial path TP.1→PAGE.1 →TEAM.1.) Also, the candidate predicates for e_1 and e_2 have the common prefix path expression (TP.1→PAGE.?→TEAM.1) corresponding to the path. Therefore, in order to associate a player with the team logo which appears on the same page, $Share_{TP.1 \rightarrow PAGE.? \rightarrow TEAM.1}(o_1, o_2)$ is derived.

(2) V-Association Predicate: The system searches for combinations of candidate predicates which have annotations including the same example object value. If it finds such a combination and the object variables of the predicates are o and o', the system derives $o \stackrel{v}{=} o'$.

For example, look at annotations of candidate predicates on o_1 and o_3 in Expression A. Because the example objects have the same value, the system derives $o_1 \stackrel{v}{=} o_3$.

Ex. Johnson in TP	Ex. T in TP	Ex. Johnson in PP	Ex. ProfileJ in PP	Ex. Johnson in BS	Ex. 0.304 in BS	Ex. Johnson in VD	Ex. J1 in VD
Johnson	T	Johnson	ProfileJ	Johnson	0.304	Johnson	J1
Johnson	T	Johnson	ProfileJ	Johnson	0.304	Johnson	J2
Thomas	T	Thomas	ProfileT	Thomas	0.320	Thomas	T
Larry	G	Larry	ProfileL	Larry	0.315	Larry	L1
Larry	G	Larry	ProfileL	Larry	0.315	Larry	L2

Figure 22 Target relation.

5.4. CREATION OF THE NESTED STRUCTURE

Creation of the nested relation reflecting the specified grouping structure is straightforward. It depends on the position of examples and repetition marks (*) on the Canvas. Figure 23 shows the grouping specified on the Canvas shown in Figure 15. The nested relation satisfying the requirement is constructed by applying Projection and Nest operators (Fischer, 1983)[5]. In this example, the following expression creates the nested relation shown in Figure 24.

$$Nest_{V=(\boxed{J1})}(\pi_{\underline{Johnson},\underline{0.304},\boxed{J1},\textcircled{T},\underline{ProfileJ}}(TR))$$

Figure 25 shows the result of mapping the nested relation into the Web structure consisting of the index HTML page and SMIL pages. Figure 4 corresponds to the result SMIL page for Johnson.

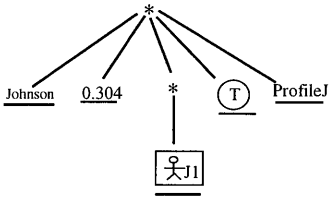

Figure 23 Grouping specification.

Ex. Johnson in TP	Ex. 0.304 in BS	V Ex. J1 in VD	Ex. T in TP	Ex. ProfileJ in PP
Johnson	0.304	J1 J2	T	ProfileJ
Thomas	0.320	T	T	ProfileT
Larry	0.315	L1 L2	G	ProfileL

Figure 24 Result nested relation.

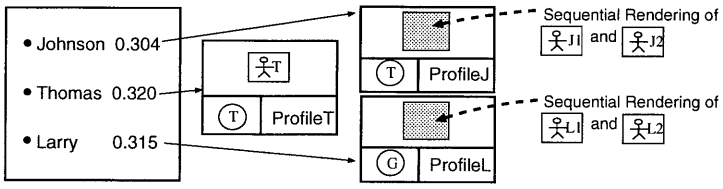

Figure 25 The result of data operation.

6. RELATED WORK

To the best of our knowledge, the visual user interface proposed in this paper is the first one that attains seamless integration of authoring, querying, and restructuring of data for multimedia view construction. However, there are several systems which support users in specifying presentation of the set-at-a-time data manipulation result. Delaunay^{MM}(Cruz et al., 1998) provides

a visual user interface where the user can drag and drop graphical icons. The icons are used to represent the multimedia data objects that are the result of queries from different information sources. However, the user has to enter SQL-like queries in a visual format. In particular, it uses WebSQL (Mendelzon et al., 1996) as the target language for Web queries. RBE (Kishnamurthy et al., 1995) also allows users to drag and drop various GUI widgets for the purpose of rendering data stored in a database. In RBE, explicit utilization of *domain variables* is required to specify how to connect data in the database to the widgets. Moreover, the source data of rendering in RBE is assumed to be well-structured data. (In fact, the source data in (Kishnamurthy et al., 1995) is a single relation.) Tiramisu (Anderson et al., 1999) is a web-site management system where presentation of the synthesized Web pages can be specified with authoring tools such as FrontPage. (They refer to this specification process as *implementation.*) In Tiramisu, querying and restructuring of the underlying data have to be done with *the site schema*, quite a different operational flamework from that for implementation tools. The basic idea underlying Tiramisu is that logical design of a web-site should be independent of its presentation. But we believe that the idea does not necessarily imply that the operational flameworks for logical design and presentation design have to be different. SuperSQL (Toyama, 1998) and SQL+D (Baral, 1998) are query languages which integrate display specification into SQL queries. They provide no means to represent queries visually.

There are a number of authoring tools for HTML and SMIL pages. Basic design of our visual user interface has been influenced by DreamWeaver (Macromedia, Inc), where various data objects can be arranged by drag-and-drop operations. FrontPage (Microsoft Corp.) allows users to incorporate SQL queries into special documents named ASP. But it has no concept of "Example" for the set-at-a-time manipulation. Also, there are some SMIL authoring tools such as RealProducer G2 Authoring Tool (RealNetworks, Inc.). As far as we know, there is no SMIL authoring tool which provides querying facilities.

The concept of "Example" was first introduced in QBE (Zloof, 1977). QBE was designed for relational databases where data has flat structure. Construction of nesting structures is supported by languages such as STBE (Özsoyoglu, 1989) and RBE (Kishnamurthy et al., 1995). They all assume that data has explicit and regular structure, which is not guaranteed in the context of semistructured data. Recently, several visual query languages for semistructured data have been proposed. DataGuide (Goldman et al., 1997) gives an abstract specification of semistructured data. It can be used as a query language with examples. XML-GL (Ceri et al., 1999) is a graph-based query language for XML documents. HQBE (Kitagawa et al., 2000)(Morishima et al., 1999(a)) can construct various views over the Web, RDBs, and structured documents. Also, there are a number of graphical query languages for various information systems (Chavda et al., 1997) (Kuntz et al., 1989), some of which support query formulation with drag-and-drop. All these languages request the user to *specify* queries according to some metadata such as database

schema. In contrast to this, our framework *infers* queries from instance-based example operations. In this sense, our approach is similar to the query by example approach in the information retrieval context (Flickner et al., 1995) (Ishikawa et al., 1998). The difference is that our flamework has to infer the intended *data operation* rather than distance-based (similarity-based) queries.

Another important issue in the multimedia presentation context is that the presentation often needs to satisfy some constraints (Jourdan et al., 1998) (Özsoyoglu et al., 1996). For example, a data object may request a specific accompanying data object when it is involved in the presentation. Also, presentation of particular combinations of objects may be required to be synchronized. Such constraints may be given by users or may be obtained from the underlying data semantics. Although the development of our interface has not been focused on this aspect, insights on such issues could be incorporated into our framework.

7. CONCLUSION

In this paper, we have proposed a visual user interface which amalgamates authoring, querying, and restructuring functions in construction of multimedia Web views. By introducing the concept of designating existing data objects as examples into the drag-and-drop-based operational framework, the interface allows the user to seamlessly integrate object-at-a-time and set-at-a-time operations. The proposed scheme can also cope with semistructured data which often appears in the context of multimedia Web view construction. The interface allows the binding of examples to be decided dynamically according to the user's interaction. We have provided the formal semantics of the data operation through the interface.

A prototype system to implement the proposed scheme is under construction. Experimental evaluation of the interface is an important future research issue. We believe that in the context of InfoWeaver the interface is far more user friendly than the existing interface, but we have to verify this by quantitative analyses. Also, it is important to clarify rules which meet the user's intention in practical situations. Another important research issue is analysis of the expressive power of the framework based on the formal semantics. These issues will be discussed in forthcoming papers.

Acknowledgments

The authors are grateful to members of KDE group, Database Laboratory, University of Tsukuba for the inspiring discussion. This work was supported in part by the Ministry of Education, Science, Sports and Culture, Japan, and Information Broadcasting Laboratories, Inc. Thanks are also due to Nippon Television Network Corporation for providing sample video data.

Appendix: Derivation of Predicates for Target Sets (Case 2)

Derivation of Clue Predicates $Clue\text{-}Pred_{cl}(c)$ is derived when the user invokes 'Clue' operation for an example object. (We call it *the target example* here.) The system determines $Clue\text{-}Pred_{cl}(c)$ using $Candidate\text{-}Pred_e(o)$, $path(e)$, $path(cl)$, and the condition on c ($cond(c)$) specified in the ClueBox.

1 Let P be the predicate '$p[cond(c)]$' where p is $path(cl)$. For example, consider Operation (8) in Subsection 4.4. Then, P =BS.1→PAGE.1→ TUPLE.1→ AVG.1[$c > 0.3$]. Note that P holds if and only if (1) c is the clue object cl itself, which the user specified in the 'Clue' operation, and (2) $value(c)$ is greater than 0.3.

Also, let P_e be the candidate predicate for the target example e. (That is, $Candidate\text{-}Pred_e(o)$.) In Operation (8), P_e = BS.1→PAGE.?→TUPLE.1 →P-NAME.1[o⟨Johnson⟩].

2 Find the longest prefix partial path expression p_e of P_e such that $path(e)$ and $path(cl)$ share the same partial path in the range of p_e. In Operation (8), p_e = BS.1→PAGE.?→TUPLE.1. Then, the clue predicate is derived from P by replacing the corresponding prefix part of P with p_e. Therefore, $Clue\text{-}Pred_{cl}(c)$ becomes BS.1→PAGE.? →TUPLE.1→AVG.1[$c >$ 0.3]. The result predicate holds if and only if (1) $path(c)$ conforms to the path expression, and (2) c satisfies the condition ($c > 0.3$). In the above example, the predicate holds if and only if c is a hitting average over 0.3.

3 If the system cannot find p_e, the user interface informs the user that it is the wrong operation because the system cannot infer the relationship between the example e and the clue cl.

Derivation of S-Association Predicates $SA\text{-}Pred_{e,cl}(o, c)$ is used to connect an object o that satisfies $Candidate\text{-}Pred_e(o)$ with another object c that satisfies $Clue\text{-}Pred_{cl}(c)$. It holds if and only if o and c have the same relative position as e and cl. As a result, the target set is constrained to contain only those objects that satisfy $Candidate\text{-}Pred_e(o)$ and have corresponding objects that satisfy $Clue\text{-}Pred_{cl}(c)$. The S-Association Predicate is derived as follows.

1 Find the longest prefix partial path expression p_e of $Candidate\text{-}Pred_e(o)$ such that $path(e)$ and $path(cl)$ share the same partial path in the range of p_e. In Operation (8), p_e = BS.1→PAGE.?→TUPLE.1. Note that p_e is a partial path expresssion of $Clue\text{-}Pred_{cl}(c)$ as well because of its derivation procedure.

2 Then, $SA\text{-}Pred_{e,cl}(o, c)$ becomes $Share_{p_e}(o, c)$. That is, if o and c share the prefix path in the same way as e and cl, the system infers that c serves as a clue for o. In Operation (8), $SA\text{-}Pred_{e,cl}(o, c) = Share_{BS.1 \rightarrow PAGE.? \rightarrow TUPLE.1}(o, c)$.

Notes

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

2. *value*(*e*) is used to derive V-Association. We explain details in Subsection 5.3.

3. The annotations are used only in the derivation process. They are removed after the derivation process is completed.

4. In query languages for semistructured data such as Lorel (Abiteboul et al., 1997), *path variables* are used to represent this association. For example, A.1→B.?→C[o_1]∧ A.1→B.? →D[o_2 > 3]∧$Share_{A.1 \rightarrow B.?}(o_1, o_2)$ would be translated into A.1→B.?{R_1} →C[o_1]∧ A.1→B.? {R_2} →D[o_2 > 3]∧$R_1 = R_2$ where R_1 and R_2 are path variables. However, in this paper, we adopt the former style in order to clearly separate the predicates.

5. If multiple repetition marks appear at the same hierarchy level, the order of applying Nest operators must be specified by the user, since Nest operators are not commutative.

References

Abiteboul, S. (1997) Querying Semi-Structured Data. *Proc. 6th International Conference on Data Theory (ICDT'97)*, pp. 1-18.

Anderson, C. R., Levy, A. Y., and Weld, D. S. (1999) Declarative Web-Site Management with Tiramisu. *Proc. ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*

Abiteboul, S. et al. (1997) The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, Vol. 1, No. 1, pp. 68-88.

Baral, C., Gonzalez, G., and Son, T. C. (1998) Design and Implementation of Display Specifications for Multimedia Answers. *Proc. ICDE'98*, pp. 558-565.

Buneman, P. (1997) Semistructured Data. *Proc. 16th ACM Symposium on Principles of Database Systems (PODS'97)*, pp. 117-121.

Ceri, S., Comai, S., Damiani, E., Fraternali, P., Paraboschi, S., and Tanca, L. (1999) XML-GL: A Graphical Language for Querying and Restructuring XML Documents. *WWW8 / Computer Networks*, Vol. 31, No. 11-16, pp. 1171-1187.

Cruz, I. F. and Lucas, W. T. (1998) Automatic Generation of User-Defined Virtual Documents Using Query and Layout Templates. *Theory and Practice of Object Systems*, Vol. 4, No. 4, pp. 245-260.

Chavda, M. and Wood, P. T. (1997) An ODMG-compliant Visual Object Query Language. *Proc. VLDB'97*, pp. 456-465.

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

Flickner, M. et al. (1995) Query by Image and Video Content: The QBIC System. *IEEE Computer*, Vol. 28, No. 9, pp. 23-32.

Fischer, P. C. and Thomas, S. J. (1983) Operators for Non-first-normal-form Relations. *Proc. IEEE COMPSAC83*, pp. 464-475.

Goldman, R. and Widom, J. (1997) DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. *Proc. VLDB '97*, pp. 436-445.

Ishikawa, Y., Subramanya, R. and Faloutsos, C. (1998) MindReader: Querying Databases Through Multiple Examples. *Proc. VLDB '98*, pp. 218-227.

Jourdan, M., Layaïda, N., Roisin, C., Sabry-Ismaïl, L., and Tardif, L. (1998) Medeus, and authoring environment for interactive multimedia documents. *Proc. ACM Multimedia'98*, pp. 267-272.

Kuntz, M. and Melchert, R. (1989) Pasta-3's Graphical Query Language: Direct Manipulation, Cooperative Queries, Full Expressive Power. *Proc. VLDB'89*, pp. 97-105.

Kishnamurthy, R. and Zloof, M. (1995) RBE: Rendering By Example. *Proc. ICDE'95*, pp. 288-297.

Kitagawa, H., Morishima, A., and Mizuguchi, H. (2000) Integration of Heterogeneous Information Sources in InfoWeaver. *Advances in Databases and Multimedia for the New Century - A Swiss/Japanese Perspective -*, World Scientific Publishing (to apper).

Macromedia, Inc. Macromedia Homepage. http://www.macromedia.com/.

Microsoft Corporation. Microsoft Homepage. http://www.microsoft.com/.

Morishima, A. and Kitagawa, H. (1999(a)) 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. (in Japanese)

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

Morishima, A., Kitagawa, H., Mizuguchi, H., and Koizumi, S. (2000(a)) Dynamic Creation of Multimedia Web Views on Heterogeneous Information Sources. *Proc. Thirty-Third Hawaii International Conference on System Sciences (HICSS-33).*

Mendelzon, A. O., Mihaila, G. A., and Milo, T. (1996) Querying the World Wide Web. *Proc. PDIS'96*, pp. 80-91.

Özsoyoglu, G., Hakkoymaz, V., and Kraft, J. D. (1996) Automating the Assembly of Presentations from Multimedia Databases. *Proc. ICDE'96*, pp. 593-601.

Özsoyoglu, G., Matos, V., and Özsoyoglu, Z. M. (1989) Query Processing Techniques in the Summary-Table-by-Example Database Query Language. *ACM TODS*, Vol. 14, No. 4, pp. 526-573.

RealNetworks, Inc. RealNetworks Home Page. http://www.real.com/.

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

Toyama, M. (1998) SuperSQL: An Extended SQL for Database Publishing and Presentation. *Proc. SIGMOD '98*, pp. 584-586.

Wiederhold, G. (1992) Mediators in the Architecture of Future Information Systems. *IEEE Computer*, pp. 38-49.

W3C (1998) Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. W3C Recommendation, http://www.w3.org/TR/REC-smil.

Zloof, M. M. (1997) Query-by-Example: A Data Base Language. *IBM Systems Journal*, Vol. 16, No. 4, pp. 324-343.