Integrated Framework for the Visualization of
Relational Databases and Related Web Content

Tsuyoshi Sugibuchi, Yuzuru Tanaka
Hokkaido University Meme Media Laboratory
{buchi, tanaka}@meme.hokudai.ac.jp

Abstract
This paper proposes an integrated framework for relating and visualizing relational databases and related Web content. This framework allows users to dynamically extract relations from HTML documents over the Web, to relate them with relations stored in local databases, and to interactively visualize them. The emphasis in our framework is put on integrating visualization schemata, source data, Web content and Web applications by means of the relational algebra. Our framework is based on the principle that every visualized object represents a view relation. In our framework each visualization schema is treated as an extended view relation, and can be related with local database records by applying relational join operations. Furthermore, our framework provides users with a method for specifying special view relations to be extracted from HTML documents over the Web. These “Web views” can be combined with database views through relational database operations to simultaneously visualize mutually related database content and Web content.

1. Introduction
The WWW is quite a mountain of data and services. If we can flexibly and dynamically extract some significant portions of these data and services, and dynamically integrate them with a database visualization system, we can start with a visualization process with local database relations, and variously extend our visualizations by combining them with related Web content objects.

Our visualization framework is based on the relational algebra and the query modification method. Any visualizations and their operation are treated as view relations and their modifications. In addition, our framework provides an interactive way of designing and coordinating database visualizations. Moreover, following the basic concept of our framework, users can define a special view relation by extracting a set of mutually related content objects from HTML documents distributed over the WWW. We call such view relations Web views. A Web view may take, as its attribute values, not only static content objects in HTML documents but also parameters of functions and services of Web applications.

2. Database Visualization Framework
2.1. Basic example
Figure 1 shows a simple database visualization example created by our visualization system. In this example we visualize the periodic table of elements. Our system provides a set of operator components for composing database queries and visualization schemata as a set of three-dimensional visual functional components called box. By combining these three-dimensional components, users can interactively compose queries and visualizations. In this example, the TableBox (a) specifies a relation to visualize, and the TemplateManagerBox (b) specifies the use of a tiny ball as a template to represent each record. The OriginBox
In the first visualization result (A), atoms are represented by small balls, and geometrically arranged according to their melting points, boiling points, and densities. Users can obtain new visualizations using this visualization as a source data set. A SelectBox (e) selects some visualized atoms enclosed by its shape. These selected atoms are associated with a big ball representation (f), overlaid with a whole set of atoms (g), and then visualized like (B). This composition performs the brushing-and-linking manipulation. Whenever users move or resize the SelectBox (e) to select a new range of atoms, the visualization (B) immediately updates its contents. Thus our framework allows users to construct visualizations incrementally and to derive more specialized visualizations from preceding visualizations.

2.2. Basic principle of our approach

In our framework, all visualizations and manipulations are defined based on relational algebra. Our framework is based on the principle that every visualized object represents as a view relation. Each visualization generated by our framework corresponds to a view relation. Similarly a selected region in a visualization is a view relation. Deriving new visualization from preceding one is equivalent to applying relational operations on the corresponding view relations.

2.3. V-extended views

As mentioned in section 2.2, a visualization is treated as a view relation. We call this view a V-extended view. Figure 2 shows an example of a V-extended view. V-extended views define graphical specifications of visualizations and associations between source data and graphics. In a visualization process, V-extended views are created on demand, and modified dynamically to combine the corresponding visualizations.

A schema of a V-extended view consists of a set of VD (visually derived) attributes. Each VD attribute corresponds to a property of graphics, like color, position, orientation, size, etc. In the example illustrated in Figure 2, the V-extended view consists of four VD attributes, TEMPLATENAME, XCOORD, YCOORD, and ZCOORD. The attribute TEMPLATENAME specifies a template. A template is a visual object for representing each retrieved record. Users can design a template by combining arbitrary graphical components. If a template includes a component which has some parameters to represent attribute values, VD attributes corresponding to these parameters are added to a schema of a V-extended view. In this example, VD attribute TEXT are added in the schema of this v-extended view, because this template includes a StringBox for displaying some text as a label. Each template is referred to by its name, and the VD attribute TEMPLATENAME specifies the name of the template for representing records. Other attributes XCOORD, YCOORD, and ZCOORD specify the geometrical locations in 3-dimensional space to arrange visualized records. Therefore, this schema of V-extended view in Figure 2 defines a 3-dimensional scatter plot chart with labeled balls.

To visualize a relation we need to associate a source data relation with a visualization schema. This process is called a visual mapping, which, in our framework, is performed by mapping a source relation to a V-extended view. Each attribute of a source relation is associated with a VD attribute of a V-extended view. In the example in Figure 3, the source data attribute SYMBOL is associated with the VD attribute TEXT to display the symbol for each atom. In this case, values of the attribute SYMBOL are the values of the attribute TEXT. We can also associate some computed values with a VD attribute. Three source data attributes E_GROUP, PERIOD, and DENSITY are associated with three VD attributes XCOORD, YCOORD, and ZCOORD of this V-extended view. Values of these three attributes are...
processed to adjust the origin and the range of the coordinate system and to normalize each record's location in the 3-dimensional space. To represent each retrieved record by using the "BALL" template, values of the attribute TEMPLATENAME are set to "BALL".

The specification of a v-extended view can be described as a query written in SQL as shown in Figure 2. In this query, the visual mapping is defined in the select list, and the conditions for restricting visualized records are specified in the WHERE clause. Our framework treats the specification of a visualization as a query modification.

2.4. Operator components

Our framework provides a set of operator components for users to specify visualization schemata and visual mappings.

Table 1 Operator components

<table>
<thead>
<tr>
<th>Operator components for specifying representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>TemplateManagerBox</td>
</tr>
<tr>
<td>AxisBox, OriginBox</td>
</tr>
<tr>
<td>ContainerBox</td>
</tr>
<tr>
<td>OverlayBox</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator components for specifying queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableView</td>
</tr>
<tr>
<td>SelectBox</td>
</tr>
<tr>
<td>RecordFilterBox</td>
</tr>
<tr>
<td>JoinBox</td>
</tr>
</tbody>
</table>

Table 1 shows types of operator components. In this table, names of operators have a similar form like "xxBox" because our current implementation of this framework is based on a 3D component-ware system IntelligentBox [11] and operator components are developed as functional components called boxes. Operator components are classified into two categories, operators for specifying representations and operators for specifying queries.

Users can connect such operators together to specify various visualizations. The current implementation of our framework provides two methods for composition. First, by using plugs and jacks users can connect components as if they connect AV components. Second, users can drop an operator component into another component to connect these two components.

Logically, each operator component has some procedure to modify v-extended views. Operator components have several ports for inputting or outputting v-extended views. Each component takes v-extended views as input and applies its relational operations to input views, and outputs modified views. For example, Figure 5 shows a view modification procedure of a TemplateManagerBox. In this figure, v-extended views are represented as SQL queries. This TemplateManagerBox adds the TEMPLATENAME attribute to the select statement list to specify a template to represent records. Each operator component takes SQL queries specifying v-extended views as input, rewrites them to apply its operation, and outputs the modified queries.

Users can specify a visualization just by combining operator components. The whole system specification of a visualization is represented as a flow diagram consisting of operator components and connections among them. We call this diagram a query flow diagram. Figure 6 shows an example query flow diagram. A ContainerBox evaluates the input SQL queries and retrieves records.

2.5. Visualized object as a view relation

In our framework, any portions of visualization correspond to some view relations. Figure 7 illustrates a visualization process starting with the visualization result obtained in Figure 1. In this example visualization (B) corresponds to a v-extended view (i), and each ball representing an atom corresponds to a small v-extended view with the single tuple representing this ball. Any portion of the visualization corresponds to a relational view with only those tuples representing balls in this portion. Such portions working as v-extended views can be joined with other relations.

In this example a user picks a ball representing ruthenium from this visualization, and drops it into a SelectBox. This dropped ball is associated with a view relation (ii) illustrated in Figure 7. SelectBox extracts this view relation. The output view relation can be joined with a relation storing data of major isotopes of each atom. This operation
is defined by using a JoinBox. As a result, a ContainerBox visualizes isotopes of ruthenium as the visualization result (C).

In this section, we extend our visualization framework to integrate database content with related Web content. We adopt the relational model as the basic data structure. We introduce virtual view relations called Web views as relations of Web data extracted from the Web. A Web view is used to extract a relation from HTML documents over the Web. Users can combine some information extracted from the Web with local database relations by specifying Web views and joining them with local relations.

Figure 8 shows an example Web view. The specification of a Web view includes structural matching patterns to select some portions of HTML documents, and rules for wrapping selected portions to obtain a relational view schema. We call the specification of a Web view an extraction path. In our current implementation, we use a XPATH-like notation to describe extraction paths. This notation has three little extensions from the original XPATH notation.

Our notation uses a special expression to specify branched patterns for matching. In this example, first and second <TD> nodes appearing in each row of a table are selected. Attributes or texts of HTML nodes included in selected portions are associated with attributes included in a relational view schema. For this purpose, the description of each extraction path includes extraction points. In this example the text enclosed by the first <TD> node is extracted as the attribute name. The attribute src of <IMG> node is extracted as the attribute image. Moreover, our notation uses backreferences to define more complicated extractions.

Figure 9 shows an example of extracting information from the World Heritage List page presented by UNESCO. This page includes the name of each nation as a section heading. These headings are followed by text lines each of which includes the inscribed year and the name of a world heritage site. Moreover, each of these lines is linked to a detailed information page of the heritage. A Web view for extracting a relation from these pages is specified by the WebViewBox. A WebViewBox is a special operator that picks up a relation from HTML documents over the Web. Users can combine some information extracted from the Web with local database relations by specifying Web views and joining them with local relations.

Figure 7 Visualizing details on demand

3. Integration with related Web content

In this section, we extend our visualization framework to integrate database content with related Web content.

3.1. Web views

To import extracted Web data into our framework, we adopt the relational model as the basic data structure. We introduce virtual view relations called Web views as relations of Web data extracted from the Web. A Web view is used to extract a relation from HTML documents over the Web. Users can combine some information extracted from the Web with local database relations by specifying Web views and joining them with local relations.

Figure 8 shows an example Web view. The specification of a Web view includes structural matching patterns to select some portions of HTML documents, and rules for wrapping selected portions to obtain a relational view schema. We call the specification of a Web view an extraction path. In our current implementation, we use a XPATH-like notation to describe extraction paths. This notation has three little extensions from the original XPATH notation.

Our notation uses a special expression to specify branched patterns for matching. In this example, first and second <TD> nodes appearing in each row of a table are selected. Attributes or texts of HTML nodes included in selected portions are associated with attributes included in a relational view schema. For this purpose, the description of each extraction path includes extraction points. In this example the text enclosed by the first <TD> node is extracted as the attribute name. The attribute src of <IMG> node is extracted as the attribute image. Moreover, our notation uses backreferences to define more complicated extractions.

Figure 9 Visualizing world heritages

Figure 8 A Web view and its extraction path
component for extracting relations from the Web by evaluating extraction path descriptions.

In this example a local relation NATIONS is visualized at first. The GDP per capita of each nation is visualized by using an expandable bar, then nations included in Europe or north Africa are selected by the SelectBox. The Selected portion is joined with the Web view, and photographs of world heritage sites in selected nations extracted from the Web and visualized as shown in Figure 9.

3.2. Implementation

We have implemented our system based on a component-ware system IntelligentBox [11] for developing three-dimensional interactive applications. The IntelligentBox system architecture provides 3D visual components called boxes. Boxes are interactive and functional components, and have slots as a logical interface for accessing internal states of each box. End users can specify slot connections between two boxes to define interoperation between them. We have implemented operator components as boxes and used the slot connection mechanism to connect them.

4. Related research works

The visualization system by Livny et al. [8], called DEVise, is most closely related to ours. This system uses the relational algebra not only to associate source data to visualization schemata, but also to specify relationships among multiple visualizations. RVN (Relational Visualization Notation) [6] is another work which focused on using the relational model to associate tabular data to visualizations.

Chris North classified types of coordinations among different types of visualizations, and he developed a visualization system, called Snap-Together-Visualization [10]. Our framework supports multiple visualization coordination in a similar manner to DEVise's. In our case two visualizations may be related through relational operations. Another approach is to associate visualizations to users manipulations tightly. In this approach a visualization content must be updated smoothly according to a users manipulation for changing a parameter value. Dynamic querying is the general term for this approach. Our framework has not supported these techniques yet. VQE [5] supports both multiple view coordination and dynamic queries.

In our framework visualizations can be specified by composing flow diagrams. The dataflow model is one of the popular models for visual programming environment, and has been adopted in many interactive visualization systems, e.g. AVS [4], DataExplorer [3], and Tioga-2 [1]. In particular, Tioga-2 is closely related to our work. From the viewpoint of visual query language, our framework is related to Kaleidoquery [9]. In Kaleidoquery, a user's querying process is modeled as a filter flow diagram regarded as a set of refinement steps of retrieved information.

5. Concluding remarks

In this paper we have proposed a new framework which provides an integrated environment for visualizing both database records and Web content objects. In our framework, each visualization corresponds to a view relation, and is specified by a database query. Users can create such queries by interactively combining operator components. In addition, we have extended this framework to support extractions of relations from HTML documents over the Web. Users can extract relations from the Web by specifying Web views, and combine them with local database relations to visualize databases with the visualizations of related Web content objects. We have developed a database visualization system based on this framework, and shown some visualization examples created by this system.

References