JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 9, NO. 6, JUNE 2006(pp. 720-727)

Integrated Methods of Various Media Generators in The SuperSQL Query Process System

Sang-Gyu Shin[†], Tai-Suk Kim[††], Motomichi Toyama[†††]

ABSTRACT

In this paper, we propose a method which allows the SuperSQL query processor to share as much code as possible among various generators, each of which is responsible for the output of a certain medium. SuperSQL is an enhanced query-processing system that converts database records into a variety of formats such as XML, HTML, PDF and etc. However, the existing SuperSQL media generator would require creation of a different processor for each medium, causing duplicated development cost. This research makes three main contributions: First, it analyzes the structures of various media, examining any possibility of integration based on their common structure. Second, it also facilitates the addition of a new output media generator by separating constructors and decorators from each medium. Last, it provides an integrated user interface to each media by method of the Media Abstraction Table Concept. We also show the performance and feasibility of our system using experimental results.

Keywords: Database, Database Publish, Multimedia Publish, Integration

1. INTRODUCTION

As the amount of information available to us is literally increasing before our eyes, the value of data as an organizational asset is widely recognized. To get most available information, users require tools that simplify the tasks of managing data such as relational databases. A database system is a central repository of information, which is shared among a number of applications. It is shared not only by multiple applications but also by multiple types of business applications. For example, the same data is used to produce printed reports, spreadsheets, web pages or XML documents to exchange data over the Internet.

A conventional query language for a relational database system yields another relation and results in a flat table form. So called 4GL systems such as "report writers" have been used to translate the information obtained from a DBMS to a specific application. Unfortunately, there is no standard language that covers the specification of such translations into various types of application data.

SuperSQL[1-4] is an extension of SQL that allows query results to be presented in various media for publishing and presentations with simple but sophisticated formatting capabilities. SuperSQL queries can generate various kinds of materials, for example, a LaTeX source document to publish query results in a nested table, HTML or XML source documents to present the result on WWW browsers, and other media including MS-Excel spreadsheet, PDF, etc.

In this paper, we restructure the parts of the media generator in the SuperSQL query processor in order to efficiently support target publishing media.

※ Corresponding Author : Tai-Suk Kim, Address : (614-714) 995 Eomgwangno, Busanjin-gu, Busan, Korea, TEL : +82-51-890-1707, FAX : +82-51-890-1724, E-mail : tskim@deu.ac.kr

Receipt date : Feb. 28, 2006, Approval date : June 7, 2006

[†] Dept. of Information and Computer Science, Keio Univ. Japan
(E-mail : shin@db.ics.keio.ac.jp)
[††] Dept. of Computer Software Engineering, Dongeui Univ.
[†††] Dept. of Information and Computer Science, Keio Univ. Japan
(E-mail : toyama@db.ics.keio.ac.jp)

An old problem of the SuperSQL system is that it increases of media generation by adding target media to publishing. This means that individual target media have their own media processing parts to publish media. These processing methods are very inefficient in the SuperSQL media generating processor.

The new system we propose in this paper is based on a trinity data model that is composed of data, structure, and media abstraction. Using these concepts, we extract common processing parts from each media generator, and make those characteristics common to all. We then present methods for integrating media generators into a common programming interface using the trinity data model. Finally, we present an experimental study of the new system using a workload of publishing for each target media document.

The rest of the paper is organized as follows. In section 2 we provide a brief overview of SuperSQL and in Section 3 describe our new system architecture and proposed methods to integrate media generators and how to use it for publishing media documents. Section 4 discusses related work and compares our system with related research. Section 5 reports some experimental results and Section 6 concludes the paper.

2. AN OVERVIEW OF SUPERSQL

SuperSQL extends SQL with TFE to generate various kinds of structured publishing and presentation documents. Fig. 1 shows the system architecture of SuperSQL.

TFE is an extension of a target list in SQL. Unlike an ordinary target list, which is a comma-separated list of attributes, TFE uses new operators (connectors and repeaters) to specify the structure of the document generated as a result of the query. Each operator is associated with a dimension respectively: horizontal (first dimension), vertical (second dimension), and depth (third dimension). Several kinds of special functions are also supported.

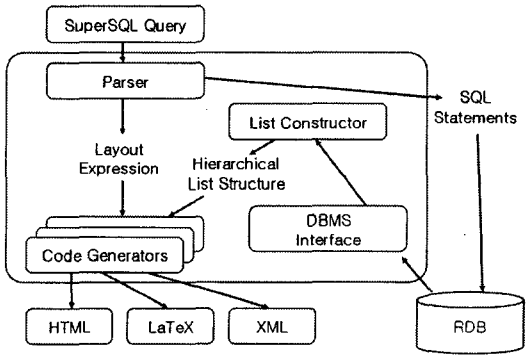


Fig. 1. Architecture.

We adopted the GENERATE clause with TFE in place of the SELECT clause. The GENERATE clause if made to target media definitions and TFE definitions by extended SELECT clause. In addition to this, the media into which the query results convert to has to be specified as below.

GENERATE *<medium><TFE>*
FROM *<form clause>*
WHERE *<where clause>*

As a *<medium>* specification, we introduced HTML to specify the generation of an HTML source file in this paper.

3. TRANSLATION SYSTEM

The problem with the previous version of the SuperSQL query processor is that a new code generator is needed for each target medium when a new target media is added. In order to improve the redundant structure of media generator processors, we have redesigned the system architecture based on the trinity data model (Fig. 2) concept. Every published medium has its own constructors (tag set, Medium Abstraction on Fig.2), data stream (Value on Fig. 2), and nested structure (Structure on Fig. 2), while relational tables do not. Thus, in converting from tables in a relational database to

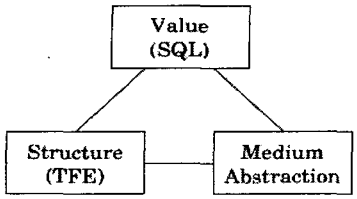


Fig. 2. The Trinity Data Model.

target media documents, constructed information (constructors and structure) has to be added somewhere along the way. Our approach is to add constructors and to make separate structure processing separated by each object method.

The goal of the Trinity Data Model concept (TDM) is to reconstruct independently from a specific target medium. The translation system is maintained as just one base structure for generating a target medium. Some specific target media use this integrated base structure by assigning the tag definition methods in the processor for constructing specific media. To achieve this goal, we restructure the translation system as an object-oriented system. The advantage of this concept is that the SuperSQL becomes more extensible. We also want to provide a more efficient and common interface for programmers who want to develop a new media document using the SuperSQL system.

3.1 The Architecture of Proposal System

Our system consists of three parts: DBMS Interface, constructors Generator and Media Generator based on the Trinity Data Model. Fig. 3 shows the architecture of our system.

DBMS Interface: Value part of the TDM. A common SQL sentence that is extracted from SueperSQL through the parser, retrieves and obtains the result from existing database systems.

Constructors Generator: Media abstraction part of the TDM. This part makes basic information to generate target media. It is composed of a TFE Processor, Constructor Processor, Function Processor and Decorator Processor.

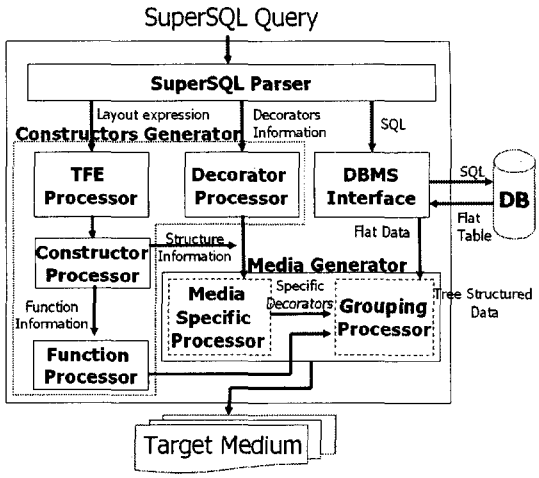


Fig. 3. The Proposal System.

Media Generator: This is composed of a Media Specific Processor, Grouping Processor, etc. The basic information to generate target media made by a constructors generator and is used in this part. A Media Specific Processor executes the specific processing as does an HTML link option.

3.2 Making Meta-data as XML

The SuperSQL query is separated into a common SQL query and a TFE expression by the parser. The common SQL query is passed to DBMS and a flat query output is obtained. It is converted into a hierarchical list structure by a list constructor in accordance with the nesting structure of the layout expression. However, this hierarchical list is just a grouped list that does not have any information about TFE or any decorative operator information for a specific target medium.

The grouping process is the most important feature of SuperSQL. As we see below, a SuperSQL query makes a table such as can be seen in Fig. 4.

```
Generate HTML [P.age, [P.name, [c.model, [C.color]!
                                M.maker]!]!]!
From people P, car C carmaker M
Where P.cid = C.id and C.mid = M.id
```

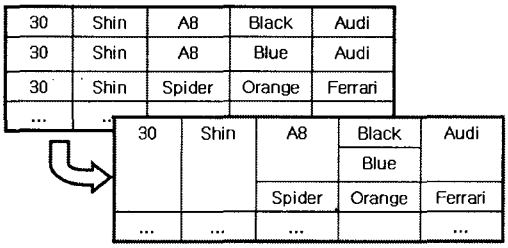


Fig. 4 Grouping in SuperSQL.

To make a grouped table as in Fig. 4, we first need to know how to group, then into which medium to convert the original table grouping information and other information for the target media. One simple way to do this is to add TFE information to the intermediate result from the relational database. In our approach, this would be done by making the XML[10] document as shown in Table 1. Note that the XML type has TFE information (G1, C1, etc) to in order to show grouping information.

```
Query : GENERATE HTML [A % [ B, C, D ]! ],
        FROM items
Layout Expr.  : (G1 (C3 1 (G2 (C1 2 3 4))))
```

Our translation system groups the flat result table of a relational database based on grouping information in a layout expression. The number of Layout Expr. represent column numbers of results from the relational database. Commas (,), exclamation points (!), percents (%), and brackets ([ ]) are expressions in TFE for connecting and repeating the flat result table. Layout expression is

Table 1. Transforming Query Output.

Hierarchical List	((A1 ((B11 C11 D11) (B12 C12 D12))) (A2 ((B21 C21 D21) (B22 C22 D22) (B23 C23 D23))))
XML Type	<G1><C1><COL1>a1</COL1><G2> <C1><<COL2>B11</COL2><COL3> B11</COL3>

created by the SuperSQL query parser. Our system translates the result of DBMS to an XML type file as shown in Table 1. Fig. 5 shows the XML type schema. This XML schema is used to add constructors and decorators for creating target media.

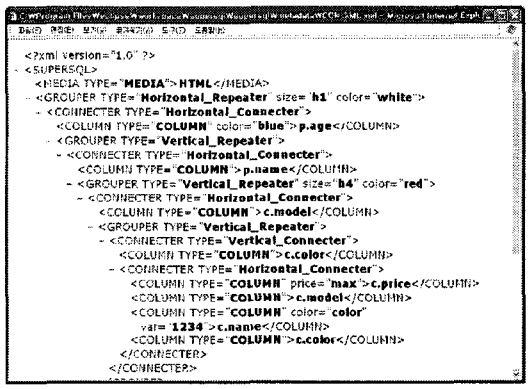


Fig. 5. View of Created XML.

3.3 Process for Construction

In order to publish RDB query results to a target medium, we must start with the intermediate data, which is a XML document. To achieve this, we get the processor to check the TFE types and to handle these TFEs. Fig. 6 shows the process for a TFE type check.

```
Process Matching TFE Expression Type
Input: meta-data base on XML doc. list L
Let Gs be the element of L
For L is not ended do:
    Let Ts be the string parsed Gs for check TFE type;
    If Ts equals G1 type then;
        Invoke G1 method with the result and L
    If Ts equals G2 type then;
        Invoke G2 method with the result and L
    If Ts equals G3 type then;
        Invoke G3 method with the result and L
    If Ts equals C1 type then;
        Invoke C1 method with the result and L
    If Ts equals C2 type then;
        Invoke C2 method with the result and L
    If Ts equals C3 type then;
        Invoke C3 method with the result and L
    Else: // Ts is unknown type
        Print the error message;
```

Fig. 6. Matching Process for TFE Expression type.

This processor checks the TFE type, invoking the most appropriate method for the result and some information for processing the constructor. The expressions G1, G2 and G3 are known as three dimensional repeaters meaning horizontal, vertical and depth, respectively and C1, C2 and C3 are the three dimensional connectors. Each TFE method which has received data makes an applicable structure by referencing the specific tags of an applicable medium.

In order to work independently from a specific medium, the data type constructors (specific tags) are defined in a medium generator independent from structure processing. For example, Fig. 7 shows the constructors required to publish LaTeX and HTML documents.

As shown in Fig. 7, we basically need to define a preamble, separator and trailer in order for the medium document to be structured properly. To separate these specific constructors from structure processing, we design the structure processing parts to they can add (control) these tags separately from the program source. Fig. 8 shows the basic processing with regard to the structure processing part for TFE.

LaTeX

Dim	Preamble	Separator	Trailer
0	\documentstyle{article} \begin{document}		\end{document}
1	\begin{tabular}{cc}	&	\end{tabular}
2	\begin{tabular}{c}	\\	\end{tabular
3		\vfill\eject	

HTML

Dim	Preamble	Separator	Trailer
0	<html><body>		</body></html>
1	<table><tr><td>	</td><td>	</td></tr></table>
2	<table><tr><td>	</td></tr><tr><td>	</td></tr></table>
3	<a href="	">	

Fig. 7. Sample Medium Abstraction Matrices.

Process Basic Algorithm for TFE Processing System
Input: Database result by list L
Let Gs be the element of L start point at L is Sp
Let Ws & We be the starting wrap this function;
Let C be the string mean connector;
Let F be the functions type of SperSQL;
Function tfeG1(start tag Ts)
  Let Te be the end tag made from Ts, result list R
  For L is not ended do:
    If Gs equal Te them:
      Add We and C at R
      Escape process and throw R to recurred function
    Else:
      If this process works at first then:
        Add Ws at R
      Else: Add Ws and C at R
    If Ts equals G types then:
      R ←add result from process Gtype methods(Ts)
    If Ts equals Ctypes then:
      R ←add result from process Ctype methods(Ts) ;
    If Ts equals F types then:
    R ←add result from process *function*methods(Ts) ;
    Else: // Ts is unknown type
      Throw R to Error process;
return R

Fig. 8. Basic Concept Algorism of TFE Methods.

This process checks the structure type and result list R. After the check, the processor throws the TFE type and adapted point to the each structure processor.

3.4 Automata Theory for Construct

For a detailed description of proposal processes, we refer to the automata and computation theory[11]. Before we describe the automata theory, Let us first discuss methods used in our theory. We define the grammar and deterministic finite automata. Context-free grammar G can be defined as a 4-tuple:

$G = (V_t, V_n, P, S)$

- V_t is a finite set of terminals
- V_n is a finite set of non-terminals
- P is a finite set of production rules
- S is an element of V_n, the distinguished starting non-terminal

The definition of non-deterministic finite automaton (NFA) is: let Q be a finite set and let Σ be a finite set of symbols. Also let ∂ be a function from $Q \times \Sigma$ to 2^Q, q_0 be a state in Q and A be a subset of Q. We call the elements of Q a state, ∂ the transition function, q_0 the initial state and A the set of accepting states. Then a non-deterministic finite automaton becomes a 5-tuple $<Q, \Sigma, q_0, \partial, A>$. The basic process of TFE (Fig. 8) and the basic process of the tag set (Fig. 10) can be defined theoretically as laid out below. It is based on Fig. 7, Sample Medium Abstraction Matrices. The preamble is defined as Co and Separator as Cs, Trailer is defined to be Cc, and then basic pattern can be defined as a regular expression $R_B = ((C_O)^+ ((C_O)^+ ((R_B)^* \mid D^?) (C_C)^+ (C_S)^?)^*$. This grammar G_B is defined as

$$G_B = (\{E, C_O, C_C, C_S, D\}, \{Str., Obj., \lambda\}, P, E)$$
$$P = \{E \rightarrow C_O\ D\ C_C\ C_S,\ D \rightarrow E \mid Str. \mid Obj. \mid \lambda,$$
$$C_O \rightarrow Str. \mid Obj. \mid \lambda,\ C_C \rightarrow Str. \mid Obj. \mid \lambda,$$
$$C_S \rightarrow Str. \mid Obj. \mid \lambda\}$$

Str. is a result stream that is processed through a translation process. Before describing further define elements. Open tag represents T_O and name space represents N_S, Tag name is T_N, Attributes represent A, Close tag is T_C. The regular expression R_C is defined as

$$R_B = ((T_O)(N_S)^? (T_N)(A)^* (T_C)^?)^+$$

The grammar of R_B is defined as below

$$G_C = (\{C, T_O, N_S, T_N, A, T_C\}, \{Str., Obj., \lambda\}, P, E)$$
$$P = \{C \rightarrow T_O\ N_S\ T_N\ A\ T_C,\ A \rightarrow C \mid A. \mid \lambda,$$
$$T_O \rightarrow Str. \mid Obj., T_N \rightarrow Str. \mid Obj. \mid \lambda,$$
$$T_C \rightarrow Str. \mid Obj. \mid \lambda,\ N_S \rightarrow Str. \mid Obj. \mid \lambda\}$$

We can define the NFA M_B and M_C based on G_B and G_C.

$$M_B = (Q_B, \{C, D\}, \partial, q_{ini}, C_C)$$
$$Q_B = (q_{ini}, C_O, C_C, C_S, Data)$$
$$M_C = (Q_C, \{T, A\}, \partial, q_{ini}, T_C)$$
$$Q_C = (q_{ini}, T_O, T_N, T_C, N_S, Att.)$$

Our translation processor uses this parsed information before constructing target media.

4. RELATED WORKS AND EVALUATION

Research projects such as SilkRoute[5] and XPERANTO[6] have proposed techniques for efficiently publishing relational data as XML. Commercial database products such as SQL Server, Oracle, and DB2 also provide support for publishing relational data as XML[9]. However, they are limited only to a specific document medium format.

Bickmore[7] has proposed web page filtering technology and re-authoring for mobile users using Web-page data. Other research for efficient web browsing on handheld devices using page and form summarization is being studied by Buyukkokten[8]. Chen[12] proposed detecting web page structure technology. However, these researches are limited to web page data.

Another area of database publishing is applications that use databases to manage large amounts of information. To publish these data, they need to assign an operator to design the layout form.

Our work differs from the above research in the following respects. First, we support publishing to various media based on the SuperSQL. SuperSQL has features like arbitrarily nested expressions and ordered grouping, which were not supported in previous research. Second, we use Target Form Expression (TFE) which extends SQL expression. TFE expression can generate multi-level, multi-page documents immediately to keep the latter up to date. Our work proposes more efficient integration methods and offers convenience to programmers based on the SuperSQL system.

5. EXPERIMENTAL RESULT

In this section we report the performance of the proposed integration methods. We compare the processing time among varying table. The performance of our system is compared among example tables with tuple numbers varying from 10000 to 40000. Our proposed method improved execution time by 45% compared with existing systems without an integrated processor. Table 2. shows the amount of source code compared with the existing systems. Every medium generator has at least a 90% decrease in the amount of source code. The experimental result shows the efficiency of our integrated method compared to existing systems. Fig. 9 shows the published XML and HTML results.

Table 2. Number of Each Generator.

	compare system	proposal system
parser	3500	4000
XML	2400	190
HTML	1800	170
LaTeX	1800	170

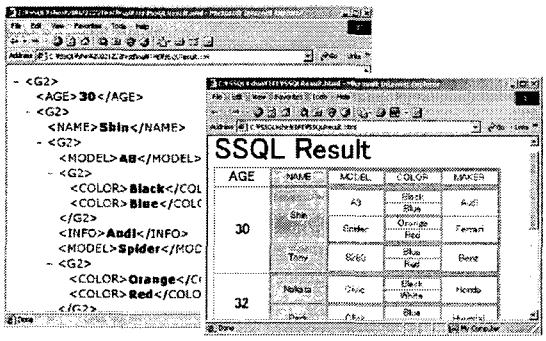


Fig. 9. Media Publication Example.

6. CONCLUSION

In this paper, we have focused on the problem of the media generator in SuperSQL query processor. One of the most important issues in database research is how to use the result of databases. The research which will transform database results into media is progressing in various fields. Our research is one such fields. In this research, we proposed a system that integrated a media generator process in SuperSQL which uses SQL language extension to specify the construction of a hierarchical list from relational data. By integrating a media generator and supplying a common user interface in this manner, our application could reuse existing functions and APIs in order for users to structure a target medium from relational data sources. The bulk of this research was devoted to exploring efficient mechanisms for making grouped tree-structure meta-data in the form of XML documents. Moving toward this goal, we first present, the SuperSQL system and illustrate its problem, focusing on the media generator. In addition to this, we design our reformation system based on common parts and find out main differences between some media namely: constructors, attributes and nested structure. Our experimental results showed that use of the provided method and integration of media generators can provide a significant performance benefit and easier generating environment. This is because the user or programmer does not need to know the structure or processing function thanks to utilization of common method.

7. REFERENCES

[1] M. Toyama, "SupserSQL:An Extended SQL for Database Publishing and Presentation," *Proceedings of the ACM SIGMOD International Conference on management of Data*, pp. 584-586, 1998.

[2] M. Akahori, T. Arisawa, and M. Toyama, "Data Integration on Relational Database and XML Using SueprSQL," *IPSJ Transactions on Databases*, Vol. 42, No. SIG8, pp. 66-95, 2001.

[3] M. Sasada and M. Toyama, "Generating Dynamic Presentation for Database Contents Using Sequencing Operators of SuperSQL," *IPSJ Transations on Databases*, Vol. 46, No. SIG13, pp. 65-77, 2005.

[4] S. G, Shin, T. Arisawa, and M. Toyama, "The Integration of Media Generators in SuperSQL Query Processor," *Proceedings of the Third International Conference on ELPIIT*, pp. 72-76, 2003.

[5] M. F. Fernandez, Y. Kadiyaska, D. Suciu, A. Morishima, and W. C. Tan, "SilkRoute: A framework for publishing relational data in XML," *ACM Transactions on Database Systems*, Vol. 27, No. 4, pp. 438-493, 2002.

[6] M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian, "XPERANTO : A Middleware for Publishing Object-Relational Data as XML Documents," *Proceedings of the International Conference on Vary Large Database*, pp. 646-648, 2000.

[7] T. W. Bickmore, A. Girgensohn, and J. W. Sullivan, "Web Page Filtering and Re-Authoring for Mobile Users," *The Computer journal*, Vol. 42, No. 6, pp. 534-546, 1999.

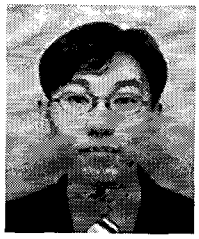
[8] O. Buyukkokten, O. Kaljuvee, H. Garcia-Molina, A Paepcke, and T Winograd, "Efficient web browsing on handheld devices using page and form summarization," *ACM Transaction on Information Systems*, Vol. 20, No. 1, pp. 82-115, 2002.

[9] C. Gould, Z. Su, and P. T. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications," *Proceedings of the International Conference on Software Engineering*, pp. 645-654, 2004.

[10] M. Nicola and J. John, "XML parsing : a threat to database performance," *Proceedings of the International Conference on Information and Knowledge Management*, pp. 175-178, 2003.

[11] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley Publishers, Boston, MA., 2000.

[12] Y. Chen, W.-Y. Ma, H.-J. and Zhang, "Detecting Web page Structure for Adaptive Viewing on Small Form Factor Devies," *Proceedings of the International WWW Conference*, pp. 225-233, 2003.



Sang-Gyu Shin

received the BE degree in computer engineering form Dongeui University, Korea, in 2000 and the MS degree in OPEN and environmental system from Keio University, Japan, in 2003. He is currently a PhD candidate in the same University. His research interests include database systems, XML, data mining. He is a student member of the ACM and IEEE.

Tai-Suk Kim

received the B.S. degree in Electronic Engineering form Kyungpook National University, Korea, in 1981 and the M.S. and ph.D. degree in Computer Science from KEIO University, Japan, in 1989 and 1993, respectively. Since 1994, he has been a faculty member of the Dongeui University, where he is now Professor in department of Computer software engineering. His research field has been in information system, internet business, network game and NLP.

Toyama Motomichi

received the PhD degree in administration engineering from Keio University in 1992. He is a assistant professor in the Graduate School of Information and Computer Science at Keio University. His research interests include database systems and database publishing. He is a member of IEEE Computer Society and ACM.