Automated SuperSQL Query Formulation Based on Statistical Characteristics of Data

Jun Nemoto and Motomichi Toyama

Keio University, Yokohama, Japan
{jun, toyama}@db.ics.keio.ac.jp
http://www.db.ics.keio.ac.jp/

Abstract. In this paper, we propose an automated SuperSQL query formulating method based on statistical characteristics of data so that the end-users can obtain the desired information more easily without burden of web application development side. SuperSQL is an extension of SQL to generate various kinds of structured presentations and documents. In our method, first, web producers prepare a provisional SQL query and send it to a relational database. Next, the automated algorithm formulates a SuperSQL query based on statistical information of the query result. Finally the resulting SuperSQL query is executed to return the categorized table to end-users. Experimental results demonstrate that the implemented system enables web producers to construct data-intensive web sites, which have better structure with respect to the recognition of the data by end-users.

1 Introduction

Data-intensive web sites such as shopping sites and news sites have become increasingly popular. Upon the presentation of database contents on these pages, it is common and popular to add a search box or sort key buttons to ease a user's effort.

In this paper, we introduce a mechanism to incorporate categorizing capability so that the users can obtain the desired information more easily. For example, in an online bookstore, you can quickly find the book you want if all the books are categorized by either authors, publishers, price ranges and so on. However, it is much more difficult for web producers to prepare categorized layouts, depending on the attribute in which users are interested. In this paper, in order to reduce such burden of web application development and to provide information using the dynamically arranged layout, we propose an automated SuperSQL query formulating method based on statistical characteristics of data. SuperSQL is an extension of SQL with the TFE which stands for Target Form Expression to generate various kinds of structured presentations and documents.

The outline of our processing flow is shown in Fig. 1. In our method, (1) web producers prepare a provisional SQL query and send it to a relational database to gather necessary information. (2) The automated algorithm formulates a SuperSQL query based on statistical information of the query results. Our current

K.V. Andersen, J. Debenham, and R. Wagner (Eds.): DEXA 2005, LNCS 3588, pp. 44–54, 2005.

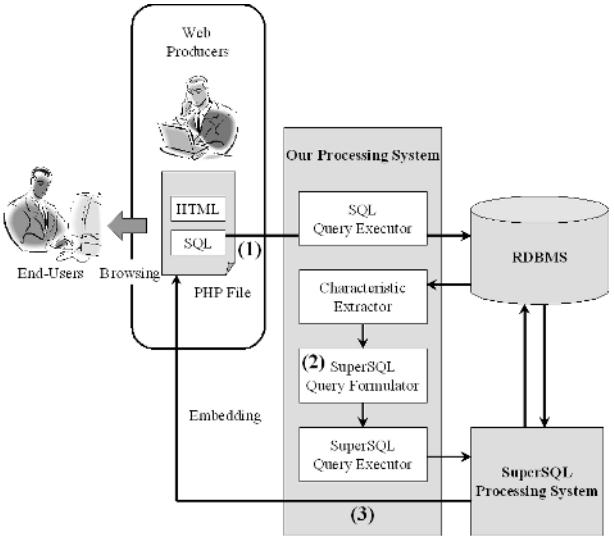


Fig. 1. Outline of System

implementation basically uses the number of distinct values. As for numeric type data, we use the coefficient of variation. (3) The resulting SuperSQL query is executed to return the categorized table to end-users.

The rest of the paper is organized as follows. An overview of SuperSQL is presented in Section 2. In Section 3, statistical characteristics of data which we assume are defined. Section 4 describes an automated SuperSQL query formulating method based on it. Section 5 shows implementation of our method and experimental evaluation. The related work is presented in Section 6 and the conclusion is given in Section 7.

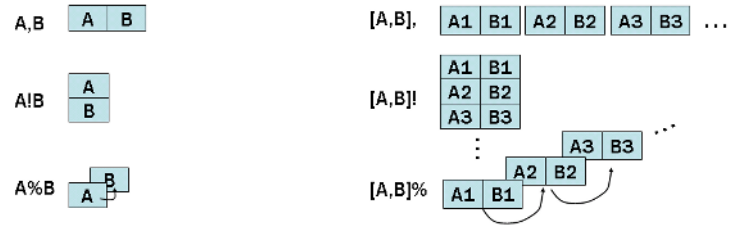
2 SuperSQL

In this section, we briefly describe SuperSQL, which is the query language to return the final results to end-users.

2.1 An Overview of SuperSQL and TFE

SuperSQL is an extension of SQL with the TFE which stands for Target Form Expression to generate various kinds of structured presentation documents [5][6].

TFE is an extension of target list of SQL. Unlike an ordinary target list, which is a comma-separated list of attribute, TFE uses new operators, which are connectors and repeaters to specify the structure of the document generated as the final results of the query. Each connector and repeater is associated with a dimension: when generating a Web document, the first two dimensions are associated with the columns and rows of `<table>` structure of HTML and the third dimension is associated with hyper-links.



(a) Connectors (b) Repearters

Fig. 2. Connectors and Repeaters in TFE

We have introduced the `GENERATE <medium> <TFE>` clause to SQL syntax to clarify the distinction with `SELECT <target list>` clause. Other target medium designations, which are allowed in the current implementation but not treated in this paper, include XML, Excel, LaTeX, PDF, etc.

2.2 Connectors and Repeaters

Binary operators represented by a comma (,), an exclamation point (!), and a percent (%) are used as the connectors of first three dimensions. Conceptually, they connect the objects generated as their operands horizontally, vertically, and in the depth direction respectively. Examples of connectors are in Fig. 2(a).

A pair of square brackets ([ ]) followed by any of the connectors is a repeater for each dimension. It will connect multiple instances in its associated dimension repeatedly. Example of repeaters are in Fig. 2(b).

2.3 Decorative Operators

Decorative operators are supported to designate decorative features of outputs, such as font size, table border, width, image directory, in the form of @ follows a TFE, and decorative specifications are in a pair of braces({}), which are separated by comma. Each decorative specification is in the form of "item = value". Decorative operators are described as below.

$$<TFE>@\{item_1 = value_1, item_2 = value_2, ..., item_n = value_n\}$$

3 Statistical Characteristics of Data

In order to formulate TFE, we use statistical characteristics of data. In this section, we describe two criteria, which we assume to use at present.

3.1 Instance Characteristics

Instance characteristics are measures to know how often distinct values appeared in an attribute. Let $A = \{a_1, a_2, \cdots, a_m\}$ denote the attribute set in the target list of SQL query. Then instance characteristics IC_i are defined as below.

$$IC_i = |distinct(V_i)|$$

Note that V_i is $\{v_{i1}, v_{i2}, \cdots, v_{in}\}$ where v_{ij} represents an instance that belongs to the attribute a_i in the query results. In addition, *distinct* is the function which eliminates duplicate values. Primarily, we assume that the smaller the instance characteristics, the much you want the table to be categorized.

However, this assumption may not always be true. For example, consider a student grade database, which has information of student's id, name, sex, department, subject, and grade. In this case, since the domain of the attribute sex is male and female, its instance characteristics must be at most two. Though this value is small one, most of end-users do not want the resulting table to be categorized by sex. Regarding these problem on the semantics of the attribute, we describe a countermeasure in Section 3.3.

3.2 Numeric Value Characteristics

If the data type of an attribute is numeric, the assumption in Section 3.1 may not be true. In case of numeric data type, end-users want the resulting table to be categorized when the attribute has many different values. Therefore, we define numeric value characteristics NC as follows.

$$NC_i = \frac{1}{\bar{v_i}} \sqrt{\frac{1}{n} \sum_{j=1}^{n} (v_{ij} - \bar{v_i})^2}$$

This measure shows the coefficient of variation of v_i. The coefficient of variation can evaluate the practical dispersion without depending on the unit. If this value is higher than the threshold, we divide the interval $[min(V_i), max(V_i)]$ into equal ranges and provide the resulting table categorized by these intervals. As long as the threshold for the coefficient of variation is properly set, we can avoid unintentionally dividing the table which has many distinct values concentrated on a narrow interval.

3.3 User Interaction

End-users often would like to browse database query results from various points of view. However, the automated layout formulation depending on statistical characteristics of data cannot always generate the categorized table by which all end-users are satisfied. Therefore, we introduced interaction with end-users.

Primary Attribute. When end-users browse data-intensive web sites, the role and importance of each attribute is not equal. For example, consider an online shopping site. An end-user may want to select the item from the list of products ordered by makers, and another end-user may want to select the item from the list of products categorized by price ranges. We call the attribute given the highest precedence as the primary attribute for the user. The formulation of TFE is affected by the primary attribute.

Selecting History of Primary Attribute. The primary attribute varies from one user to another. If we can obtain the information about the primary attribute selection, the algorithm can narrow candidates of the attribute for categorization. For example, consider again the student grade database in Section 3.1. Suppose that departments and subjects are repeatedly selected as the primary attribute, and sex is rarely selected. Then, the history of selecting the primary attribute can prevent the algorithm from selecting sex as the grouping target in the default layout.

4 Formulating SuperSQL Query

In this section, we present an automated SuperSQL query formulating method. Since we choose the HTML as output media, we focus the formulation of TFE from now on. Regarding FROM clause and WHERE clause, we basically use the one in the original SQL query.

In formulating TFE, we start from choosing the attribute which is most suitable for grouping target. In order to choose the grouping target, we define the grouping priority GP as follows:

$$GP_i = \frac{n}{IC_i}\ g_i\ m\ h_i$$

Note that g_i is the weight given for the primary attribute. If a_i is the primary attribute, it takes the parameter α, otherwise it takes 1. In addition, h_i is the weight reflecting the selecting history of primary attribute and it is the ratio which measures how often is the attribute selected as the primary attribute among all the selections. We multiply m and n to prevent the criteria from being biased by the number of attributes and tuples in the query results. We choose the attributes which have a value of grouping priority greater than the threshold, as the grouping target.

The attribute selected as the grouping target is promoted the outermost so that other attributes are nested. For example, suppose that the SELECT clause of original SQL query is:

SELECT sid, department, subject, grade

and the attribute which have a value of grouping priority greater than threshold is department. In this case, the formulated TFE is:

[department , [sid , subject , grade]!]!

Moreover, in case that subject is also the grouping target, the resulting TFE is:

[department ! [subject , [sid , grade]!]!]!

The result of the SuperSQL query with above TFE is as follows.

department1		
subject1	sid1	grade1
	sid2	grade2
	sid3	grade3
department2		
...	...	...

As the example shows, if there are more than one attributes for grouping, the attribute which has higher grouping priority is placed on outer side, so that it is treated as larger category. We connect the grouping target and other attributes horizontally, vertically, and in the depth direction in order from the inner connector.

Taking the numeric value characteristics into consideration, SuperSQL query formulation process is a bit different. First, in case that the data type of the primary attribute is numeric, we examine whether the numeric value characteristics are higher than the threshold. If it is higher than the threshold, we divide the domain of the primary attribute into equal ranges and the start value and the end value of each range is inserted into a temporary table. Then we join the temporary table and other tables and formulate the SuperSQL query, which makes the table grouped by each interval. On the other hand, if the numeric value characteristics are under the threshold, we enter into ordinary process, which calculates the grouping priority.

5 Implementation and Evaluation

5.1 Implementation

We have implemented a SuperSQL query formulating system based on statistical characteristics of data. In our current implementation, the above mentioned procedure is implemented as a PHP function. In this section, we describe the specification of the function using the running example.

Function Specification. The specification of implemented PHP function is as follows:

int **dcssql** (string *sql*, boolean *style* [, string *primaryAttribute*])

The **dcssql** function inserts SuperSQL query results into the position where the function is called. A SuperSQL query is formulated using *sql* and *primaryAttribute*. Note that both arguments are string and *primaryAttribute* can be omitted.

The argument *style* is Boolean value to specify whether to add style information to the generated SuperSQL query. Style information is specified using decorative operators in Section 2.3.

```
1   /* sample.php */
2   <html>
3   <head>
4   <title>The List of Books</title>
5   <meta http-equiv="Content-Type" content="text/html;charset=EUC-JP"/>
6   </head>
7   <body>
8   <form method="get" action="sample.php">
9   <input type="radio" name="primary" value="name">Name
10  <input type="radio" name="primary" value="author">Authors
11  <input type="radio" name="primary" value="publisher">Publishers
12  <input type="radio" name="primary" value="category">Categories
13  <input type="radio" name="primary" value="price">Price Ranges
14  <input type="radio" name="primary" value="pdate">Published Date
15  <input type="submit" value="Choose!">
16  </form>
17  <?php
18      include("dcssql.inc");
19      $sql = "SELECT b.name, b.author, b.publisher,
          b.category, b.price, b.pdate FROM book b";
20      dcssql($sql, TRUE, "$_GET[primary]");
21  ?>
22  </body>
23  </html>
```

Fig. 3. An Example of a PHP File

Example. As defined above, once a web producer gives the **dcssql** function a standard SQL query, a style flag and a primary attribute, a SuperSQL query is formulated, executed, and the resulting HTML are returned. We suppose as usual that web producers provide it to end-users, together with other PHP and HTML codes. Notice that the web producer is expected to implement an interface to select a primary attribute and give it to the **dcssql** fuction. An example PHP file, which web producers prepare, is shown in Figure 3 and screenshots displayed when end-users access the sample PHP file are shown in Figure 4 and Figure 5. In two screenshots, the threshold of the GP is set to 2.0 and 1.5 respectively.

We explain the details of the PHP source in Figure 3. An interface to specify a primary attribute is described in Line 8-16. Though it is provided by radio buttons in this example, any HTML form interface is available as long as the primary attribute can be exclusively selected.

The include function in Line 18 includes the file, which has the entity of the **dcssql** function and others.

Line 19 sets the original SQL query. In this example, we assume a simple table of books which has isbn, name, authors, publishers, categories, price, and

Fig. 4. The Result of sample.php ($\theta_{GP} = 2.0$)

published date as attributes. Note that price is numeric, published date is date type, and the others are string type. This table is assumed to contain one hundred records of books. In case that all the records in this table are returned to an end-user as query results, the instance characteristics of categories and publishers are higher than the others.

Line 20 receives form data about the primary attribute via the GET method and calls the **dcssql** function.

When end-users access the sample.php in Figure 3, the result without considering the primary attribute will be displayed. The result in Figure 4 is grouped by categories and publishers according to the grouping priority. In addtion to these attributes, the result in Figure 5 is grouped by authors. More remarkably, categories and other attirbutes are connected in the depth direction since the number of grouping targets are more than 3.

5.2 Experimental Evaluation

In this section, we present experimental evaluation of the implemented system.

We used databases of books, news, and geographic statistical information about Japan. To simplify queries, each database consists of only one table. The books table is the same database used in Section 5.1. The news table has ID, titles, text, sources, large categories, small categories, and arrival time as attributes. The text attribute is relatively large data so that the data size varies from 200 Bytes to 6000 Bytes. The statistical information table has ID, prefecture name, region name, the number of population, the area and so on. Except for the name of prefecture and region, all the attributes are numeric.

In this experiment, the threshold and other parameters are decided empirically. The threshold of the NC and the GP are set to 0.3 and 3 respectively. The parameter for the primary attribute is set to 5.

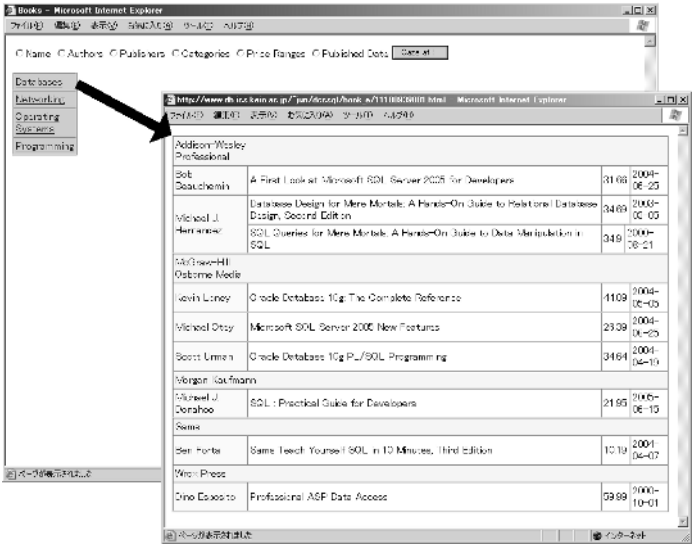


Fig. 5. The Result of sample.php ($\theta_{GP} = 1.5$)

Table 1. The Flat Table versus Generated Table

		Recognizability		Propriety of Structure	
		FLAT	NESTED	FLAT	NESTED
Books	avg.	1.50	3.75	2.38	4.25
	t-Statistic	-5.46		-6.05	
	t-Critical	1.81		1.77	
News	avg.	1.36	3.00	2.00	3.38
	t-Statistic	-3.87		-3.66	
	t-Critical	1.81		1.76	
Statistical Informaiton	avg.	2.75	3.00	3.38	3.50
	t-Statistic	-0.32		-0.18	
	t-Critical	1.77		1.77	

We got eight examinees to browse a normal flat table and the table generated by our system and then to give two scores to each table. The one score is about the recognizability and the other one is about the appropriateness of table structure. The appropriateness of structure does not mean prettiness but appropriateness for browsing the data such as books, news and statistical information. The score varies from 1 to 5, where 1 indicates the lowest level of recognizability or appropriateness and 5 indicates the highest one. Notice that all scores were given to each table as an absolute evaluation.

Then we compared the scores and performed t-tests. The null hypothesis was that there is no difference for the flat table and the categorized table generated by our system regarding the average score. The alternative hypothesis was that

the average score for the categorized table is higher than that for the flat table. All the t-tests were performed as one-tail t-tests with significance level 5%.

The experimental results are in Table 1. FLAT means the flat table and NESTED means the table generated by our system. As the tables shows, except for the case of statistical information, the absolute value of t-Statistic is greater than t-Critical about both the recognizability and the appropriateness of table structure. Therefore the null hypothesis was rejected and the alternative hypothesis was accepted. In other words, the table generated by our system has better recognizability and structure for browsing books and news than the flat table. In contrast, the alternative hypothesis was not accepted in the data of statistical information. We consider this is because almost all the attributes in the statistical information table are numeric. In short, the algorithm often formulates the TFE producing a flat table. We got similar results in case of specifying the primary attribute. However they are omitted due to the space limit.

6 Related Work

An automatic categorization technique has been proposed by Chakrabarti et al. to present relational results that are devised to minimize users' efforts in browsing [2]. In contrast to our statistical characteristics based model, it is based on cost models that are defined to estimate information overload faced by users. In order to estimate the average cost, it uses the log of queries that users have asked int the past. There shall be no difficulties to incorporate the log-based cost models to work together with our statistical approach. The other differences between that work and ours are various presentation capabilities. We provide not only a simple group-by presentation but also an arbitrary one that a TFE in a SuperSQL query allows, for example, multi-page presentation in Figure 5.

There is a research studying a dynamic SuperSQL query formulation. ACTIVIEW is an adaptive data presentation system, which creates HTML pages dynamically adapting to the size of screen [3]. Though this work is similar to ours with respect to dynamic SuperSQL query formulation, it does not consider the statistical characteristics of data in determining layouts since its main purpose is adapting the outputs to the size of various screen such as a PDA and a mobile phone equipped with a web browser.

In the context of the generation of the reports, there are many researches. QURSED is a report generating system for XML documents [4]. Differing from our approach in giving developers discretionary power to generate reports, developers can prepare various queries and layouts using the QURSED system. The Oracle report tool offers a few fixed styles and does not consider data characteristics [7]. Table Presentation System TPS is a reporting system, which can represent various layouts by several table operations [1]. Its operations are easier than usual programming effort and the QURSED Editor. However, it only provides a static structured presentation and end-users have no way to provide their preferences to change the structure.

7 Conclusion

In this paper we defined the statistical characteristics of data in a SQL query result. Then we proposed an automated SuperSQL query formulating method based on it. Experimental results demonstrate that the implemented system enables web producers to construct data-intensive web sites, which have better structure for the recognition of the data by end-users, without burden of programming.

References

1. W. Chen, K. Chung, "A Table Presentation System for Database and Web Applications", *Proceedings of IEEE EEE '04 International Conference on e-Technology, e-Commerce and e-Service*, pp. 492-498, 2004.
2. K. Chakrabarti, S. Chaudhuri, S. Hwang, "Automatic Categorization of Query Results", *Proceedings of ACM SIGMOD '04 International Conference on Management of Data*, pp. 755-766, 2004.
3. Y. Maeda, M. Toyama, "ACTIVIEW: Adaptive data presentation using SuperSQL", *Proceedings of the VLDB '01*, pp.695-696, 2001.
4. Y. Papakonstantinou, M. Petropoulos, V .Vassalos, "QURSED: Querying and Reporting Semistructured Data", *Proceedings of ACM SIGMOD '02 International Conference of Management of Data*, pp. 192-203, 2002.
5. SuperSQL: http://ssql.db.ics.keio.ac.jp/
6. M. Toyama, "SuperSQL: An Extended SQL for Database Publishing and Presentation", *Proceedings of ACM SIGMOD '98 International Conference on Management of Data*, pp. 584-586, 1998.
7. P. Weckerle, "Enterprise Data Publishing with Oracle Reports: Any Data, Any Format, Anywhere", Oracle Technology Network Technology Information 2003, http://otn.oracle.com/technology/products/reports/