A Reporting Tool Using “Programming by Example” For Format Designation

Tetsuya Masuishi
Business & Information Systems
Development Division,
Hitachi, Ltd.
890 Kashimada, Saiwai
Kawasaki, 211-8567, JAPAN
+81 44 549 1703
masuishi@bisd.hitachi.co.jp

Nobuo Takahashi
Software Division, Hitachi, Ltd.
Kaneichi Building
549-6 Shinano-cho, Totsuka
Yokohama, 244-0801, JAPAN
+81 45 826 8543
takah_nb@gm.soft.hitachi.co.jp

ABSTRACT
This paper describes a report tool in which report formats are designated by "Programming by Example"-like operations. Users specify a sample layout of an example row of relational table data on a sheet, and select an iteration pattern of the sample layout. The tool extracts a set of general formatting rules from the sample layout. The rules consist of absolute positions of non-iterative data, relative positions of iterative data, the iteration pattern, and the increment of the iteration. The tool interprets the rules and generates new reports of the format for different table data.

Keywords
Reporting Tool, Relational Database, Programming by Example, User Interface

1. INTRODUCTION
Data warehouse technologies have enabled centralized data management for decision support and planning applications. As many decisions are made by the centralized database, many formats of reports are printed for the decision makers.

Many end-users, sometimes even decision makers themselves, in planning offices have to design the report formats, since the formats represent the view of the data and cause considerable effects on the planning decision.

Such reports in Japan for decision making is a kind of reconstructed table style reports which include instance text strings and numerical data, while most of such reports in the states include graphs which already implies analysis results. The reason of the difference might be the difference of their cultures.

The reconstruction is needed because relational tables in practical use have usually many columns. We have assumed the data are stored in a relational database. The reconstruction is applied to place data of many columns to fit in the width of the paper. It is difficult to grasp reports that exceed both of the width and the height of the paper. Since the number of rows of tables are usually big enough to exceed the height of the paper, we need to reconstruct the relational table to make reports of many pages in one direction which we can handle easily. Figure 1 shows an example of a Japanese style reconstructed table report. Japanese reports usually include many ruled lines to separate adjacent data, rather than tabulation spaces. The reason is maybe the fact that Japanese words are not separated by spaces.

Many software packages called “reporting tools” support such reconstruction. Reporting tools have usually a scripting language to make programs. A program generates a report of a fixed format by embedding various data of a relational database.

This paper describes user interface of a report tool, which designates formats of reports. The user interface is designed in a “programming by example” manner [1][2]. A user creates a sample report for example table data. The tool extracts the implied formatting rules. The tool interprets the rules to generate reports reading relational table data as input. The extraction process is deterministic and does not include any

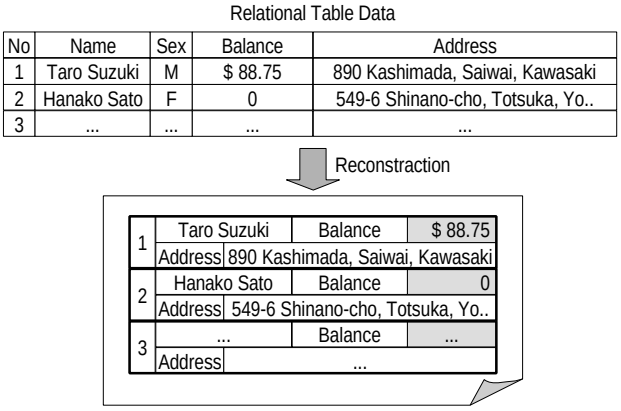

Figure 1 Typical Japanese Reconstracted Report

statistical recognition or learning process.

Sugiura and Koseki [3] shows a data entry system from e-mail text into a database. The system generates a macro program that reads e-mail text, extracts data from the text and inserts the data into database. A generalization process is employed generate a general–purpose macro program from a history of sample

operations. Our problem is easier because the input is a structured table in a database, not like e-mail text in natural language. Instead, our system employs a simple generalization process for just one example.

Myers [4] uses just one example for text formatting. The generalization process includes parsing the example text special words (ex. “chapter”, “section”, and “Appendix”), numbers, separators, and decorations (ex. lines and boxes). The parsing process encounters ambiguity. Our problem is well structured enough to make the system deterministic and easy-to-use practically.

2. SYSTEM OVERVIEW

2.1 Phases and Users

The reporting tool is used in two phases:

- Format making phase, and
- Report generating phase.

The two phases can be done by different users, and some users may do just the latter phase.

2.1.1 Making Formats

In this phase, a user makes a format not a report. Conventional reporting tools do not require table data for making formats, but this tool requires table data for us to make a sample report. (Users anyway need table data even for usual tools when they “debug” their formats.) The tool extracts a set of formatting rules and stores them in a persistent file called a format file.

2.1.2 Generating Reports

There are other users who generate reports by specifying a format file and table data. Table data can be specified by a file name if the data is stored in a file like in CSV (Comma Separated Value) file format, or by a set of retrieval statements like select statements in SQL (Structured Query Language) for a connected relational database.

2.2 System Configuration

The tool consists of:

- Format editor, which enable users to edit interactively to make format files, and
- Report generator, which enable users to generate reports by specifying a format file and table data (Figure 2).

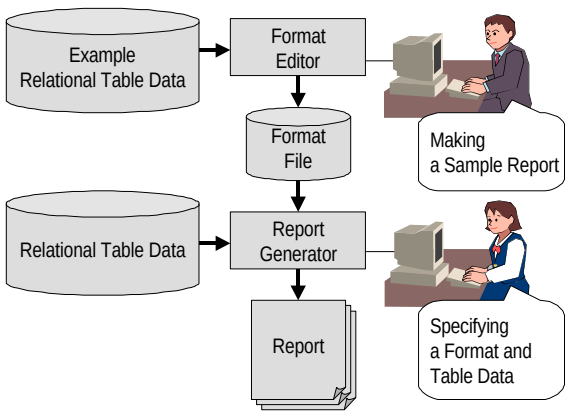

Figure 2 System Configuration

2.2.1 Format Editor

The format editor is an interactive editor for a user to make a sample report using a set of relational table data, to extract a set of formatting rules, and to save them into a format file. The editor reads a format file and example table data.

2.2.2 Format File

A format file is a conventional file that includes a set of formatting rules in our proprietary file format.

2.2.3 Report Generator

Report generator is a program that requires a format file and a set of relational table data and that generates a new report of the specified format.

3. USER INTERFACE OF FORMAT EDITOR

3.1 Programming by Example

We have employed a “Programming by Example”-like user interface for the format editor. A users makes a sample report to make formatting rules and does not have to specify general formatting rules directly. The set of rules works as a program that generates reports for other sets of relational table data. So, the process of the format editor is a kind of “programming by example”. Since some of the rules are specified directly by the user for practical use, the process should be said as “a simple and practical version of programming by example.”

3.2 Window Configuration

The format editor consists of two main windows:

- Sheet window, which represents the sample report, and
- Table window, which shows the example table data.

3.2.1 Sheet Window

The sheet window shows the sample report as a sheet image. Editing scheme of the sheet window is the following:

- Put objects on the sheet window,
- Specify iterative objects, called “a block”,
- Specify iteration pattern for the block, and
- Specify increment of the iteration.

3.2.2 Table Window

The table window shows the example table data as a relational table image. The user can copy-and-paste table data to the sheet window.

3.3 Objects on the Sheet Window

The objects on the sheet window consist of:

- Fixed text string including column names,
- Functions,
- Ruled lines, and
- Example data.

3.4 Editing Objects on the Sheet Window

3.4.1 Text String

Any text strings can be put on the sheet window and usual text string attributes like fonts can be specified.

3.4.2 Column Names

The user can copy-and-paste useful column names from the table window to the sheet window.

3.4.3 Functions

Some built-in functions are provided like date, time and so on. They work as variables.

3.4.4 Ruled Lines

Ruled lines can be drawn on the sheet window and usual line attributes like width and pattern can be specified.

3.5 Selection of an Example Row on the Table Window

3.5.1 Example Row

When starting up the editor, a default example row, the first row, is highlighted on the table window. The user can change the example row by clicking the mouse button.

Multiple example rows can be specified for special cases. Seven example rows are necessary to make a report which shows weekly trend generated from daily table data.

3.6 Placement of Example Data

3.6.1 Row Data

Data of example row(s) on the table window can be copy-and-pasted to the sheet window. Numerical data can be put as well as string data.

3.7 Specifying Iteration

3.7.1 Specifying Iterative Objects (Block)

The system makes a report by iterating print of a specified set of objects with a fixed increment of position. The user specifies which object is iterative or not. A rectangular area that surrounds the iterative objects is called a "block". The user can specify iterative objects by specifying a block by a rectangular rubber band of the mouse.

3.7.2 Specifying Iteration Pattern

The system provides iteration patterns, some of which are shown in Figure 3. The user specifies a pattern from the menu of the provided patterns.

3.7.3 Specifying Increment of Position for the Iteration

Users can specify the increment of position for the iteration by the mouse. The user can specify the increment directly. The system can calculate the increment when the user put other data than the example rows (Figure 4).

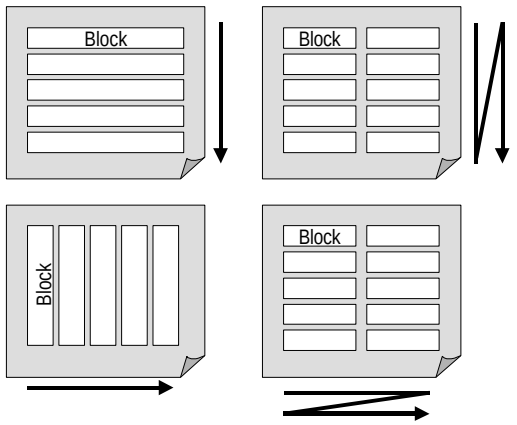

Figure 3 Examples of Iteration Patterns

3.7.4 Viewing the Image of the Sample Report

The sheet window displays the final image of the sample report, which includes iterated images of the iterative objects. The iterated images are produced by the example table data of the corresponding rows which are calculated by incrementing the row numbers of the specified example row(s) by the number of the example row(s), typically one.

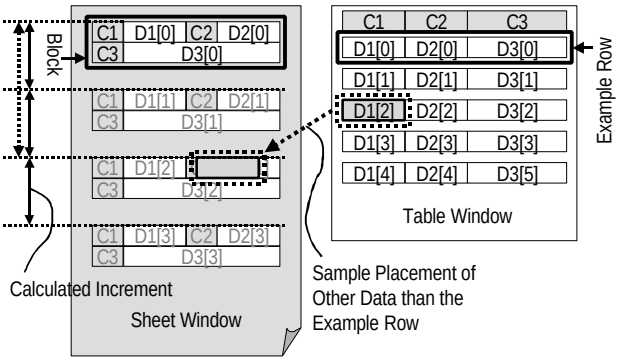

Figure 4 Calculating the Increment and the Image of the Sample Report

3.8 Adjustment

After the image of the sample report is displayed, some adjustment can be specified to create sophisticated reports.

3.8.1 Page Break and Column Break

Conditions for page break and column break can be specified. Typical condition is "when the data of the specified column change, create a new page."

3.8.2 Unified Print

When a column has repeatedly occurring data, they should be unified in some sophisticated reports like in Figure 5.

4. EXTRACTING FORMATTING RULES

4.1 Formatting Rules

The format editor stores a set of formatting rules for a sample report. Formatting rules consist of the following information.

4.1.1 List of Objects

Formatting rules include the list of objects on the sheet.

4.1.2 Block Information

Formatting rules include information about the block, including the absolute position and the size of the block.

4.1.3 Non-Iterative Objects

Formatting rules consist of the attribute of the objects and the absolute position on the sheet.

4.1.4 Iterative Objects

Formatting rules for iterative objects includes

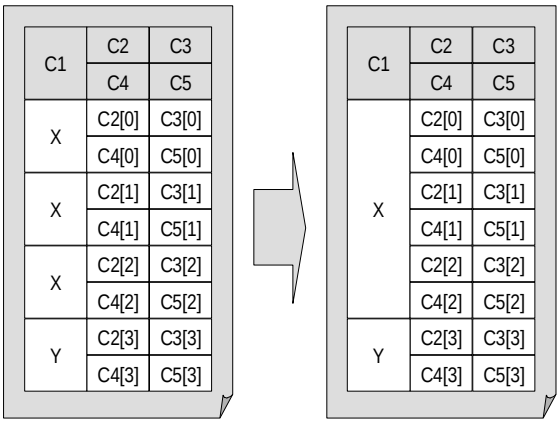

Figure 5 Unification of Repeatedly Occurred Data

- For example data

  (1) Column name,

  (2) Relative row displacement in the example table from the base of the example row, and

  (3) Relative position in the sheet from the base position of the block.

- For the other objects

  (1) Attribute of the object and

  (2) Relative position in the sheet from the base position of the block.

4.1.5 Iteration Pattern and Increment

Formatting rules include the specified iteration pattern and the increment.

4.1.6 Adjustments

Formatting rules include the specified adjustment.

5. GENERATING REPORTS

When a format file and a set of table data are given to the report generator, it generates a repot according to the algorithm, briefly described in the following.

(1) Put the entire non-iterative object on the sheet.

(2) Iterate the following sub-step for all rows of the table data.

Put all of the iterative objects at the current position with substituting the example data to the given table data at the current row of the iteration

(3) Apply the adjustment.

This algorithm generates a new report of the given format from the given table data.

6. EVALUATION

6.1 Project

We have applied the system to a real project at a Japanese company. They have developed more over than 500 formats. The applied formats includes:

(1) Reports for Planning and Decision Making

(a) Sales reports for sales persons and divisions

(b) Sales reports for products and areas

(c) Cost reports for products

(2) Mission Critical Forms

(a) Monthly reports to customers

(b) Bills

(c) In-house order forms

(3) With Images

(a) Employee files including facial photo image

(b) Price lists with product photo image

We have not been informed that they encountered any formats impossible by the format editor, except functions for detailed presentation like round corners for crossing ruled lines.

6.2 As an Example of "Programming by Example" System

We have designed the extraction process to be deterministic for practical use. The system extracts formatting rules deterministically, not statistically. Just one sample report is needed for extracting general formatting rules. To make this possible, we have some generating information been specified directly, like iteration.

This design was possible because the problem of generating reports from relational table data is well structured.

7. REFERENCES

[1] Cypher, Allen (ed.), Watch what I do: Programming by demonstration, MIT Press, 1993.

[2] Myers, Brad A., Demonstrational Interfaces: A Step Beyond Direct Manipulation, *IEEE Computer* (August 1992), 61-73.

[3] Sugiura, A and Koseki, Y., Simplifying Macro Definition in Programming by Demonstration, Proc. UIST, (1996), 173-182.

[4] Myers, Brad A., Text Formatting by Demonstration, *Proc. CHI*, (1991), 251-256.