The User Interface Is the Conceptual Model

James F. Terwilliger¹, Lois M.L. Delcambre¹, and Judith Logan²

¹ Department of Computer Science
Portland State University, Portland OR 97207, USA
{jterwill, lmd}@cs.pdx.edu

² Department of Medical Informatics and Clinical Epidemiology
School of Medicine, Oregon Health and Science University, Portland OR 97239, USA
loganju@ohsu.edu

Abstract. Frequently, the structure and description of the data in a database bears little resemblance to the structure and description of data as it appears in the tool that captured it. This makes it difficult for users to write queries because they receive little information from the database schema regarding the precise meaning of the data. We assert that the semantics of data can be more reliably understood by viewing the data in the context of the user interface (UI) of the software tool used to enter the data rather than the bare framework of a database. GUAVA (GUi As View) presents a conceptual model that captures information about user interface components. In this paper, we describe how to model a forms-based UI using a GUAVA-tree (g-tree), which can be used to generate a natural schema against which querying is simple. We then introduce and formalize the notion of a channel of database transformation operators from the natural schema to the underlying physical schema.

1 Introduction

The user interface for a typical information system comprises various forms with controls such as radio buttons, check boxes, drop-down lists, group boxes, and text boxes that allow data to be entered. The user interface usually includes labels for each control and possibly other descriptive information that helps a user understand the data that is being entered or displayed through the user interface. More than that, one form may contain a button that allows the user to launch a second form. In short, the user interface, by design, provides a detailed description of the data (to be entered and to be displayed) in a given application.

Software development environments in common use make it quite easy to assemble a user interface, simply by selecting and placing instances of the various controls. There is typically a main-memory data structure that holds the data associated with the form. This data structure is often structured as a set of tables; we refer to this as the natural schema associated with the user interface. The job of the developer, then, is to write the code that provides for any special checking or handling of the data and then stores it in a database or other persistent store. The database schema may have a structure that is quite similar to the natural schema, e.g., where there is one table for each form, with one attribute for each control on a form.
However, our focus is on database schemas where the structure can be considerably different from the natural representation. In our experience, software tools for data entry are almost always implemented with a generic schema — where the data from the user interface is stored in attribute-value pairs, and the software can be easily extended without modifying the schema. The difference between the natural schema, where attribute names are used to indicate the field, and the underlying physical database schema, where attribute names appear as data, is referred to as schematic heterogeneity. Although some effort has been devoted to providing an SQL-like query language that can handle schematic heterogeneity, the resulting language may be quite difficult for ordinary users to master.

Our research has a simple goal: use the user interface of the software tool that creates the data directly as the conceptual model for users, and allow the users to express queries against the resulting conceptual model. But we must be able to process user queries against the underlying database, as specified by the database designer. The problem, then, is how to support the natural schema for the purpose of querying, with a physical database with a significantly different structure.

This paper introduces components of the GUI As View (GUAVA) framework, as shown in Figure 1. First, the complete structure of the user interface is represented in a hierarchical structure called a GUAVA-tree (g-tree). GUAVA automatically generates a g-tree from the user interface controls based on our extensions to an integrated development environment. Next, GUAVA translates a g-tree into a simple relational table structure with a natural schema. Finally, a database designer can transform the natural schema into the underlying physical database schema using database operators. A collection of these operators form a channel that transforms the natural schema into the desired physical schema (at DB design time) and to transform simple queries from the application and the query interface from the natural schema to the physical schema (at run time).

The central purpose of this paper is to introduce the g-tree as a conceptual model and to formally define the transformation operators that can appear in the channel. The remainder of this paper is organized as follows. Section 2 provides the motivation for this work. The GUAVA framework is presented in Section 3, the main section of the paper. Section 4 describes our current work; Section 5 briefly discussed related work; and the paper concludes with a discussion of contributions and future work in Section 6.

2 Motivation

The development of GUAVA is motivated by our work with the Clinical Outcomes Research Initiative (CORI) where endoscopy reports from nearly 70 sites across the US are being compiled in a data warehouse on an ongoing basis. CORI seeks to improve the practice of endoscopy by conducting retroactive studies on de-identified patient data (i.e., the endoscopy reports). CORI develops and distributes a software reporting tool that allows the clinician to enter