QuerySheet: A Bidirectional Query Environment for Model-Driven Spreadsheets

Orlando Belo, Jácime Cunha, João Paulo Fernandes, Jorge Mendes, Rui Pereira, João Saraiva
Universidade do Minho, Portugal
{obelo,jacome,jpaulo,jorgemendes,ruipereira,ajas}@di.uminho.pt

Abstract—This paper presents a tool, named QUERYSHEET, to query spreadsheets. We defined a language to write the queries, which resembles SQL, the language to query databases. This allows to write queries which are more related to the spreadsheet content than with current approaches.

I. INTRODUCTION

Like most software artifacts, spreadsheets start as simple software systems and rapidly evolve into large and complex data-centric softwares. In such complex systems it is very important to have good support to manipulate and reason about data. Database systems use well known techniques, namely the relational model, and language support, namely the SQL language, to query, extract and reason about their data. Unfortunately, spreadsheet systems do not offer such support to query its data!

This paper presents QUERYSHEET: a tool that brings to spreadsheets the query database realm. The tool offers a query language, very similar to SQL, to query spreadsheets. This query language is based on spreadsheet models, namely ClassSheets [1], rather than in the spreadsheet data. By focusing on a simple, concise description of the spreadsheet data, rather than in a possibly large and complex spreadsheet data, we mimic the database approach: a database query writer usually reasons about the relational model of the database to express his/her queries, and not on understanding the large database. Such an approach has also the advantage of expressing queries using attribute names, and not by referring to spreadsheet areas and columns/rows numbers as provided by Google and Microsoft approaches to query spreadsheets. Both systems also require that the spreadsheet data is represented in a single matrix, that is to say that the data has to be in (or transformed to!) a non-normalized representation. In QUERYSHEET this is performed automatically by using normalization/denormalization and model inference techniques [2].

II. QUERYSHEET

In order to present our spreadsheet query language, let us consider a spreadsheet storing information about products, clients and orders. The ClassSheet model defining the business logic of this spreadsheet is shown in Figure 1 (Model worksheet). Suppose that we would like to know:

- How much have we profited from each client?
- How much have we profited from USA clients (with its histogram)?

In a regular spreadsheet system it would be very difficult extract this information from the spreadsheet data. Both Microsoft’s query system and Google’s QUERY function provide a basic form of expressing queries. However, they require users to denormalize the data (that is organized in three different tables), and to use column letters in the queries themselves. Thus, making the use of such queries complex and error-prone.

In QUERYSHEET we express the query based on the ClassSheet model, and not on the spreadsheet data. The tool provides a New Query button, that opens a text box, where the query is defined. As we can see in Figure 1 the query (expressing the first question) looks very much like SQL: it uses the same keywords and syntactic structure. Moreover, the queries use ClassSheet labels to identify the different entities involved.

When executing the query the QUERYSHEET generates the result as a ClassSheet-driven spreadsheet. In fact, two new worksheets are added to the original spreadsheet: one containing the spreadsheet data that result from the query (DATAQUERY1), and the other (MODELQUERY1) contains the ClassSheet model, as shown in Figure 2.
Figure 2 also shows the query to answer our second question. When executed, this query is applied to the results of the first one, that is to say that we can combine queries. In Figure 3 we show the (data) results of this second query.

As stated before, we need to denormalize the data into a single table (bottom-center of the figure). After we obtain the data from our model-driven environment, we begin placing the data into its denormalized state carefully grouping the correct row of information, while dealing with the problems caused by denormalized data querying such as derived data and attribute aggregation. In fact, these are well-known and well-studied problems in the database realm [5]. Our query is then translated to its exact counterpart for the QUERY function, automatically calculating the range input, and substituting the attribute names to their counterpart column letters using a lookup function, as shown in the bottom-right of the figure. Finally, we use the QUERY function to obtain the results expected. Afterwards we apply a technique introduced in previous work [2] to automatically infer a ClassSheet from the resulting spreadsheet data, as shown in the top-right of the figure.

Relevance to VL/HCC: Spreadsheets are the most popular and widely used software visual system. However, they lack a proper querying mechanism. Following the simplicity and powerful interactive interface of spreadsheets we present a tool where users can query their spreadsheet data.

Structure of the Presentation: In the demo of the QUERYSheet we will give a very interactive presentation. The QUERYSheet will be available in a laptop to motivate VL/HCC participants to use it to query their own spreadsheets. We also plan to display an accompanying poster.

REFERENCES


