Chapter 21
Readable Diagrammatic Query Language ViziQuer

Martins Zviedris
Institute of Mathematics and Computer Science, Latvia

ABSTRACT

End-user interaction with data is one of key aspects in data processing. Nowadays a lot of information systems have a custom made user interface for data input and data querying. From 1970s it is envisioned that a generic, user-friendly approach for data querying could be built, but no wide spread solution has been developed. In the paper we present a diagrammatic query language. We have done an iterative approach to design and improve the diagrammatic query language to make it user readable. Readability is analyzed with questionnaires. Readable diagrammatic query language is the first step to create a more generic and user-friendly data querying.

INTRODUCTION

Databases are keystone in modern data storage. Databases comes hand in hand with data access via query language or user-friendly interfaces that uses predefined queries. Data access and analytics is vital part of data processing. Still, there is ongoing research how to develop more users-friendly and easier-to-use data interface for domain specialist users.

Graphical query languages as an interaction level between databases and end-users have been developed from late 1970s. One of the first proposed solution was Query-by-example that was developed at the IBM (Zloof 1975). Similar approach with table-based interfaces is used other existing solutions, for example, Microsoft Access query designer.

A wide survey (Catarci, Costabile, Levialdi & Batini 1997) gives an overview of different query languages designed for SQL. The survey divides languages into three paradigms – diagrammatic, iconic and form-based. It is also possible to combine two of named paradigms together. The survey does not separate a query language syntax from the tool that supports the query formulation process for an end-users[1]. Four different approaches for query formulation in a tool are named – by schema navigation, by sub-queries[2], by matching and by range selection[3].

DOI: 10.4018/978-1-4666-8767-7.ch021

In a paper (Catarci & Santucci 1995) is given an example analysis that compares how well an end-user can write a query in a diagrammatic language QBD[4] compared to the SQL. One of main conclusions state that naïve and intermediate end-users writes queries faster and more precise in QBD rather than in the SQL. Expert users have similar results in both languages. Similar comparison of visual and textual query languages is given in (Catarci T. & Santucci G. 1995-1).

By emergence of the Semantic Web (Berners-Lee, Hendler & Lassila 2001) scientists refocused on new visual query language designs. Main feature that enable visual language redesign is that data in the Semantic Web is stored in a more semantically understandable way for end-users as the data has less technical details compared to data in relational databases.

First of all, primary and foreign keys in the Semantic Web are omitted. In our opinion, this is vital gain as in relational databases these technical terms is presented with additional column that is not easy explainable to domain end-users.

Other features includes – objects that can be contained in more than just one class, while in relational databases each object is contained in exactly one class (relation). This gives ability to define object class hierarchies in explicit manner rather than implicit via attribute values that are commonly used in relational databases design. Secondly, in the Semantic Web is no need to implement additional tables for n-to-n relations as relations. More detailed explanation between relational databases and the Semantic Web approach is available at (Barzdins, Barzdins & Cerans 2008).

As in relational databases is developed the formal query language SQL then in the Semantic Web is developed query language SPARQL (Harris & Seaborne 2013). We will not cover SPARQL in more details in this paper as there is a lot of different sources that explain SPARQL, for example, beginners tutorial (Baskauf 2014).

There are different solutions that propose a graphical notation rather than textual syntax for a query composition. For example, GRQL (Athanasis, Christophides & Kotzinos 2004), ViziQuer (Zviedris & Barzdins 2011), Tabulator (Berners-Lee et al. 2006), gFacet (Heim, Ertl & Ziegler 2010), NiteLight (Russell & Smart 2008), Gruff (Franz 2015), SPARQLgraph (Schweiger et al. 2014), OptiqueVQS (Soylu et al. 2013) or VisiNav (Harth 2010). Different approach try to utilize in the Semantic Web included semantics and generate SPARQL queries from keywords. The approach is called SPARK (Zhou, Wang, Xiong, Wang & Yu 2007).

In our opinion, one of the main drawbacks of existing approaches is that developers do not study how well end-users perceive developed query language. Languages are developed more from a viewpoint of researchers rather than end-users.

It is well known that "one picture represents 1000 words", but it is also true that a picture can be read in 1000 different ways. Thus, it is important that picture is précises and easy to read for end-users to incorporate these 1000 words in a pinpoint manner. In this paper we give a solution that shows how diagrammatic query language is developed that it can be read in a pinpoint manner by an end-user.

In our opinion, more careful studies should be done regarding to a designed graphical query language readability. A study (Juel 1988) found that there is correlation that children, who are bad readers, tend to become bad writers. We think that this is true also in a graphical query language as it is important to grasp at first language concepts by reading them, thus becoming better writers later, as the acknowledged concepts are used to write new queries. Without good concept understanding, it is hard to use these concepts later on.

Therefore, we make an assumption that – if an end-user cannot read a query in the developed language syntax then he/she will have problems later on with query composition. Thus, each graphical language