Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Systems 31 (2006) 33–56
www.elsevier.com/locate/infosys

Conversational querying

Yannis E. Ioannidis[a], Stratis D. Viglas[b,*]

[a]Department of Informatics & Telecommunications, University of Athens, Athens, Hellas, Greece
[b]School of Informatics, University of Edinburgh, Crichton Street, Edinburgh EH8 9LE, UK

Received 21 February 2003; received in revised form 16 July 2004; accepted 25 September 2004

Abstract

The traditional interaction mechanism with a database system is through the use of a query language, the most widely used one being SQL. However, when one is facing a situation where he or she has to make a minor modification to a previously issued SQL query, either the whole query has to be written from scratch, or one has to invoke an editor to edit the query. This, however, is not the way we converse with each other as humans. During the course of a conversation, the preceding interaction is used as a context within which many incomplete and/or incremental phrases are uniquely and unambiguously interpreted, sparing the need to repeat the same things again and again. In this paper, we present an effective mechanism that allows a user to interact with a database system in a way similar to the way humans converse. More specifically, incomplete SQL queries are accepted as input which are then matched to identified parts of previously issued queries. Disambiguation is achieved by using various types of semantic information. The overall method works independently of the domain under which it is used (i.e., independently of the database schema). Several algorithms that are variations of the same basic mechanism are proposed. They are mutually compared with respect to efficiency and accuracy through a limited set of experiments on human subjects. The results have been encouraging, especially when semantic knowledge from the schema is exploited, laying a potential foundation for conversational querying in databases.

Keywords: Databases; SQL; Human computer interaction; Information retrieval

1. Introduction

Suppose someone asks us the following question:

What time does CS207 start?

and after our answer a second question is immediately posed:

CS507?

We find no difficulty in interpreting the incomplete second question as:

What time does CS507 start?

In most cases, such partial questions are completed by the participants of a conversation, resulting in unambiguous new questions that they

*Corresponding author. Tel.: +44 131 650 2691; fax: +44 131 650 6513.
E-mail addresses: yannis@di.uoa.gr (Y.E. Ioannidis), sviglas@inf.ed.ac.uk (S.D. Viglas).

0306-4379/$ - see front matter © 2004 Elsevier B.V.. All rights reserved.
doi:10.1016/j.is.2004.09.002

have no trouble answering. Humans have the ability to use several factors towards completing such partial questions. Common-sense knowledge is one of them. The context of the conversation is another.

When interacting with a database management system (DBMS), we essentially start a dialogue shell with the DBMS's front end, where user and system 'speak' the same language (typically SQL). In traditional ad hoc DBMS interaction, consecutive queries are unrelated to each other, usually. Cases where the next query is an alteration of the previous one are relatively few. Therefore, it is acceptable for the user to either rewrite the whole query from scratch (if it is small), or invoke an editor and make the desired modifications.

When one considers, however, some of the applications of database technology, i.e., data mining and decision support, the situation changes dramatically. The user essentially explores the data and obtains different views or subspaces of it so that important patterns or other characteristics may be identified. This is achieved by issuing sequences of interrelated queries, which tend to be large and complicated. Rewriting every query from scratch is then out of the question, while editing the previous query becomes quite tedious and counterproductive. Whereas the user should be operating in a continuous [QUERY-ANALYZE]* cycle focusing on data exploration (Fig. 1(a)), he/she is operating in a [QUERY-ANALYZE-EDIT]* cycle (Fig. 1(b)). Instead of posing to the system the query that comes to his/her mind, the user has to enter the editor, find the appropriate places in the query text, make the necessary changes, and only then submit the new query. This forces a continuous context switch in the mental operation model of the user, reducing his/her effectiveness.

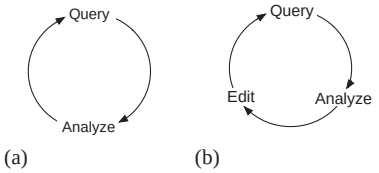

Fig. 1. The two exploration modes. (a) Uninterrupted exploration mode. (b) Exploration mode interrupted by editing.

It would be better if the human–DBMS interactions were similar to those between humans conversing, as illustrated above. The user would only have to give so much as a 'hint' to the system, just the new/different part of the query. The system would then have to understand what this hint implies and make all the necessary alterations to the original query, producing the next one. Effectively operating in the two-step cycle (Fig. 1(a)), the user's attention would thus be devoted exclusively to data exploration. We call this form of interaction with a DBMS *Conversational Querying*.

Conversational Querying is not only meaningful and desirable for textual–language interactions but for visual ones as well. First, although more pleasant than its SQL counterpart, editing of visual queries remains an interruption in the user's flow of thought, an artificial extra step in every cycle of exploration (Fig. 1(b)) that is better eliminated. Second, at the visual level, there are many human–DBMS interaction styles potentially available whose nature is such that offering them to users requires support of Conversational Querying as well. For example, consider a relation EXPERIMENT(w, x, z, y), containing the results of some experiments, i.e., containing the value of the output parameter y for various values of the input parameters w, x, and z (the composite primary key of the relation). Assume that the result of a query

```
select x,y,z
from EXPERIMENT
where (x mod 10 = 0) and (w = 3.14159)
```

is visualized as in Fig. 2(a). Whether the original query was posed in textual form as above or through some visual query tool is not relevant to this discussion. Seeing the resulting graphs, a user is quite likely to request more points in the area of x between 30 and 50, since the value of y changes dramatically there. A rather natural way to do this is for the user to indicate the area of interest *directly on the result visualization* as in Fig. 2(b). The system then takes into account the overall