Interaction Design and Architecture(s) Journal - IxD&A, N. 5-6, 2009, pp. 31-34

Towards a Dialogue Based Interface for Query Synchronization

Giuseppe Polese
Dipartimento di Matematica e Informatica
University of Salerno
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
gpolese@unisa.it

Mario Vacca
Dipartimento di Matematica e Informatica
University of Salerno
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
mvacca@unisa.it

ABSTRACT

The query synchronization is one of main issues of schema evolution: when a schema evolves the existing queries could stop working and it becomes necessary to redefine them over the new schema. In this paper, according to the recently recognized importance of advanced query formulation interfaces, the basic ideas and components underlying an interactive model of tool for supporting query synchronization are outlined.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems; H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms

Human Factors, Theory

Keywords

Dialogue, interrogative logic, metaquery, query formulation interface, query synchronization, schema evolution, schema mapping

1. INTRODUCTION

Database schema evolution naturally occurs during the life cycle of an information system: modifications of the schema can take place due to either changes in the world, or errors, or design decisions, and they can yield very significant problems. The query synchronization problem, which has been increasing its importance lately, becoming one of main issues of schema evolution, arises when a schema evolves and the existing queries stop working and it calls for the redefinition of them over the new schema. This problem, a very challenging one in the context of data warehouses, has been increasing its importance due to the development of the Semantic Web [22, 19].

Most of the current approaches to the query synchronization problem aim at rewriting queries automatically, but, unfortunately, it is not always possible and human intervention could be required. This makes the problem also relevant in the context of the man-machine interaction, both on the user and the database administrator (DBA) side: in the first case, the man-machine interaction is important to make the users able to pose queries to the database without taking care of the schema changes (as it is hoped in [15]); in the second one, the interaction should assist the DBA during the synchronization process of the queries which haven't been updated automatically.

The problem of query synchronization is very closely related to the well known problem of query rewriting in integration systems for which Clide [21], an advanced query formulation interface, has been recently proposed. Unfortunately, the problem of query synchronization differs from reformulation, as it can require approximating queries.

Our idea is the following: when it is not possible to synchronize the query automatically, the user (or the DBA) and the system should collaborate to reach an agreement about a query that the system can execute and whose answer is suitable for the user (or the DBA). This is quite natural as it follows a very common human behaviour, the dialogue for knowledge seeking: when a person asks another one for some information, it is possible that the latter is not able to answer the given question; in this case, a dialogue between the two people starts in order to state a correspondence between their knowledge.

In this paper, we investigate the possibility of developing an interactive tool that embodies the previous idea, to redefine no more working queries after schema evolution. According to our analysis, the dialogue aim is to find a schema mapping [5], while the dialogue terms are metaqueries. Therefore, the main components of the tool we envisage are: 1)a visual interface integrating query and metaquery formulation with schema mapping capabilities; 2)an interface manager (IM) for managing the dialogue between the user and the interface. We individuate the main features of the interface and we propose to use the Hintikka interrogative logic [13] for modeling the man-machine dialogue.

2. THE QUERY SYNCHRONIZATION PROBLEM

When a query does not work anymore (because of a schema change) and it is necessary to fix its definition, the best result you could expect is to rewrite the query in terms of the new schema, so that the new query has the same answer as the old one. Unfortunately, this is not always possible, and you have to be satisfied with a query whose answer is close to that of the original one (i.e. an answer that helps in knowing the whole or a part of the answer of the original query).

Example 1. *Consider the schema storing information about* *cities:*

$$S = \{Cities(Name, Country, Pop., Capital)\}$$

and the query:

$$IsCapital(x, y) \leftarrow Cities(x, y, z, t) \wedge t = \text{“}yes\text{”}$$

asking for the capitals of the countries.

Consider the schema:

$$T = \{Cities'(Name, Country, Pop.)\}$$

obtained from S deleting the attribute Capital from the relation $Cities$.

Synchronizing the query $IsCapital$ means trying to rewrite it using the information from the schema T.

According to Bernstein et al. [5], a way to solve the problem of query synchronization consists in finding a mapping between the two schemas S and T and then composing[1] the query with the mapping to obtain the new query.

Intuitively, a mapping expresses a relation between the instances of two schemas and it links the two schemas. A simple example of mapping between the two schemas of the example 1 is:

$$(\forall x, y, z, t)(Cities(x, y, z, t) \rightarrow Cities'(x, y, z))$$

In the case of the example 1, it is difficult to decide automatically how to substitute the lost information (the attribute $Capital$) so that the resulting query could be useful or suitable to the user. For instance, it could be possible to use a condition to obtain an approximate query:

$$IsCapital(x, y) \leftarrow Cities'(x, y, z) \wedge z \geq 1.000.000$$

It is obvious that different users could use different conditions.

3. STATE OF THE ART AND MOTIVATIONS

The problem of view synchronization[2] was first addressed, defined and classified in the taxonomy of *view adaption problems* by Rundensteiner, Lee and Nica, who also proposed a solution to it [22].

The existing approaches to the problem of view synchronization can be grouped in two classes, depending on the way the schema changes are performed, that is by schema change operations or using schema mappings.

Roughly speaking, a schema change operation applied to a schema produces a new version of the schema itself (e.g. delete attribute; see [1] for a taxonomy of simple schema changes). The approaches based on schema change operations [16, 2] deal with all kinds of relations between the old and new view extents, but in a limited number of cases (depending on the change operations taken in account). Therefore, extending these approaches needs a wider and comprehensive number of schema change operations to be considered. This kind of extension process has two drawbacks: first, the problem of the insufficiency of simple schema change operations has not been already solved (see [6, 17, 10] for a discussion and some proposals); second, since an algorithm for each schema change is required, the process of query synchronization isn't general.

A mapping links two schemas and, typically, mappings between two schemas S and T are described by “set of formulas of some logical formalism over (S, T)” (Fagin et al. [12], p. 999). The approaches based on the use of mappings to represent changes can handle a wider range of schema changes than those based on schema change operations. Among the mapping based ones, the approaches belonging to Generic Model Management [4, 18, 5] are more general and straightforward than the previous ones, because they don't need different algorithms for each change operation. The main limitation of these approaches is that they don't solve the problem in the case of schema changes which reduces the capacity of the source schema (like the attribute deletion of the example 1): in fact, the proposed solutions within the Generic Model Management either drop the non rewritable views (called orphans) or they exploit non-generic model semantics [4]. Always based on the use of schema mappings, the approach proposed in [15] can take in account all kinds of changes: the authors proposed to use the SchemaLog language to program the query synchronization process and they invoked the use of the cooperative query answering [9] for solving this problem in the case of capacity reducing schema changes.

Moreover, Prism [10], a mixed approach (both schema change operation and mapping based), has been recently proposed: the related tool is based on theoretical advances and it uses schema change operations, including also some compound ones, and whose meaning is represented by mappings. It reformulates SQL queries into equivalent ones (when possible), otherwise the DBA intervention is requested.

To sum up, the schema change operations based approaches are both automatic and able to deal with capacity reducing changes, but they are limited by the small number of changes and by the difficulty to extend them. The mapping based approaches either are automatic, but only reformulate the query exactly, or they are programming based and, hence, general.

Therefore, the need for a tool to support the DBA, avoiding to involve s/he in programming details, arises. In the next sections we show that such a tool can be based on the use of a visual interface.

It is also worth to note that visual interfaces are used by the schema evolution tools, but their role is limited: for example, in [10] the interface allows the DBA to write functions to synchronize queries manually; a more general use is that considered by the model management based tools, which aim to use visual interfaces to match schema versions [20].

[1]In this case the composition refers to the mapping composition operator as defined in the Generic Model Management area [5].
[2]To the best of our knowledge, the problem of breaking queries was addressed in [15].

4. THE INTERFACE BASED APPROACH TO QUERY SYNCHRONIZATION

We embrace the thesis of Lakshmanan et al. [15], according to which the user should be free to pose queries regardless of the schema changes. This means "to shield the modifications to the schema of the database from the user as much as possible."(Lakshmanan et al. [15]) and the problem becomes building a suitable "shield".

As pointed out in the previous section, the problem of query synchronization is especially challenging in the case of lost information, because it is necessary to substitute it with other ones. When the lack of information is so significant that it is impossible to rewrite the query, the problem becomes, borrowing the words from Lakshmanan et al., "How can we produce meaningful answers to queries (based on an older version of the schema) which refer to such "lost" information? " [15]. In [15] the authors suggested the application of the cooperative query answering techniques [9] in combination with the use of SchemaLog language.

We think that it is possible to go a step further, proposing a more flexible "shield" based on an human-computer collaborative approach. The observation at the base of our proposal is the following: the updating of queries is usually set only after a schema mapping has been found; in other words, the updating of queries is seen as an activity following that of a complete schema matching. This is quite unnatural if compared with human behavior: in fact, sometimes humans update their knowledge (schemata) after a question they are not able to answer and they do this through a follow-up questioning process with the questioner. Analogously, when a query is posed to a system (on an evolved schema), the system could ask for some "explanations" trying to build a bridge between its schema and that one related to the query. This kind of approach has the advantage of building (partial) mappings only if and when they are necessary.

This can be achieved through an interface having the role of realizing a dialogue in which the system and the user ask metaqueries reciprocally to build a (partial) schema mapping (i.e. a mapping only linking the metadata in the query to the old schema). In fact, as the role of the interface is to assist the user in the process of query rewriting, it should provide some help to the user: the best one is exactly the mapping, while the worst one is the possibility of browsing the whole schema. It could be also desirable to provide the user with minor suggestions; for example, when some information is lost because of a schema change, many suggestions can be provided to the user, all of them based on the application of heuristics and, as the result of heuristics is not certain, it has to be verified and this can be done submitting it to the user evaluation. In this case the user is asked a metaquery about a possible mapping.

To sum up, the interface has to allow the user either to ask the system queries and metaqueries (the first ones could help, for example, in assessing if the degree of approximation of the new query is satisfying, while the second ones help in finding the schema mapping) or to answer metaqueries, providing information about metadata and mappings.

EXAMPLE 2. *Consider the query in the example 1:*

$$IsCapital(x, y) \leftarrow Cities(x, y, z, t) \wedge t = \text{"}yes\text{"}.$$

A system trying to execute this query will receive an error message because the relation Cities no longer exists; in this case, if the system knows that the relation Cities was deleted, it could communicate this to the user [15], otherwise it could apply an heuristic (e.g. to find all the relations whose attributes are included in those of the relation Cities) and communicate the result to the user.

4.1 The features of the visual interface

Today, visual query formulation has become a common way to pose queries and metaqueries in the database field, also thanks to industrial tools like the Microsoft query builder[3] or to the researches in the field of the visual data mining [14]. Moreover, the use of visual interfaces have been recently exploited in query rewriting [21] and for schema mapping [20].

The interface we envisage has to integrate all the previously illustrated functionalities: visual meta/query formulation and visual schema mapping:

- query based dialogue

  This feature enables the user to pose queries using visual actions.

- metaquery based dialogue

  According to Ben-Eliyahu-Zohary et al. "A metaquery has the form

  $$T \leftarrow L_1, \ldots, L_m$$

  where T and L_i are literal schemes. A literal scheme S has the form $Q(Y_1, ..., Y_n)$ where all non-predicate variables Y_k are implicitly universally quantified. The expression $Q(Y_1, ..., Y_n)$ is called a relational pattern (of arity n)."(Ben-Eliyahu-Zohary et al. [3], p. 62). Using metaqueries the user can operate on schema mappings. We think to use a Clide-like interface, enriched with metaquerying capability.

- schema browsing and mapping

  This feature has to be integrated with the previous ones, in the sense that it has to be possible to use it together with them while trying to synchronize a query.

4.2 The dialogue modeling

The interface is managed by an interface manager (IM) driving the dialogue. We propose to use the Hintikka interrogative logic [13] to model the logic of the IM. The Hintikka view of reasoning can be summarized as follows: a line of reasoning is constituted by a sequence of sentences; a new sentence in such a line is either obtained by deduction or by asking an information source for it. The logic of Hintikka is modelled by a game played by the inquirer against one or more information sources, and the semantics used is the tableau method. There are two kinds of moves: *logical inference moves* and *interrogative moves.* The formers are the

[3]`http://www.microsoft.com/sql/`

typical deductive rules, and they are tableau-building rules (see [13] for a complete list). Rules for questioning serve instead to generate questions.

Therefore, the IM uses these rules to generate metaqueries to be communicated to the user through the interface.

5. CONCLUSIONS AND FUTURE WORK

In this paper, the basic ideas about the possibility of synchronizing queries in an interactive way have been presented; this is a part of a research project [8, 11], which we have been developing at the University of Salerno, aiming to build a support tool for schema evolution.

The interface paradigm we propose paraphrases a typical human behaviour: the dialogue for knowledge seeking. Analogously, the resulting schema evolution tool should interact with the user to find a (partial) schema mapping for synchronizing not working query. In order to realize this kind of process, a visual interface (with multiple functionalities), managed by an Interface Manager, is needed. Two main problems have to be still coped with: the design of the interface and the modeling of the Interface Manager. We proposed to use the Hintikka interrogative logic [13] for modeling the dialogue and we have been investigating the set of precise rules to be employed for driving the dialogue and the rules for enriching the model by heuristics. Moreover, we have been devising the set of visual actions allowing to realize the functionalities of the visual interface.

6. REFERENCES

[1] J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and implementation of schema evolution in object-oriented databases. In U. Dayal and I. L. Traiger, editors, *SIGMOD Conference*, pages 311–322. ACM Press, 1987.

[2] Z. Bellahsene. Schema evolution in data warehouses. *Knowl. Inf. Syst.*, 4(3):283–304, 2002.

[3] R. Ben-Eliyahu-Zohary, E. Gudes, and G. Ianni. Metaqueries: Semantics, complexity, and efficient algorithms. *Artif. Intell.*, 149(1):61–87, 2003.

[4] P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.

[5] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In Chan et al. [7], pages 1–12.

[6] P. Brèche. Advanced principles for changing schemas of object databases. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *CAiSE*, volume 1080 of *Lecture Notes in Computer Science*, pages 476–495. Springer, 1996.

[7] C. Y. Chan, B. C. Ooi, and A. Zhou, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2007.

[8] S.-K. Chang, V. Deufemia, G. Polese, and M. Vacca. A logic framework to support database refactoring. In R. Wagner, N. Revell, and G. Pernul, editors, *DEXA*, volume 4653 of *Lecture Notes in Computer Science*, pages 509–518. Springer, 2007.

[9] F. Cuppens and R. Demolombe. Cooperative answering: A methodology to provide intelligent access to databases. In *Expert Database Conf.*, pages 621–643, 1988.

[10] C. Curino, H. J. Moon, and C. Zaniolo. Graceful database schema evolution: the prism workbench. *PVLDB*, 1(1):761–772, 2008.

[11] V. Deufemia, G. Polese, G. Tortora, and M. Vacca. Conceptual foundations of interrogative agents. In M. Baldoni, A. Boccalatte, F. D. Paoli, M. Martelli, and V. Mascardi, editors, *WOA'07*, pages 26–33. Seneca Edizioni, 2007.

[12] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.

[13] J. Hintikka, I. Halonen, and A. Mutanen. Interrogative logic as a general theory of reasoning. In *Handbook of the Logic of Argument and Inference: The Turn Towards the Practical*, pages 295–337, 2002.

[14] S. Kimani, T. Catarci, and G. Santucci. A visual data mining environment. In S. J. Simoff, M. H. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 331–366. Springer, 2008.

[15] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. On the logical foundations of schema integration and evolution in heterogeneous database systems. In *DOOD '93*, pages 81–100, 1993.

[16] A. J. Lee, A. Nica, and E. A. Rundensteiner. The eve approach: View synchronization in dynamic distributed environments. *IEEE Trans. Knowl. Data Eng.*, 14(5):931–954, 2002.

[17] B. S. Lerner. A model for compound type changes encountered in schema evolution. *ACM Trans. Database Syst.*, 25(1):83–127, 2000.

[18] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science*. Springer, 2004.

[19] S. Melnik, E. Rahm, and P. A. Bernstein. Developing metadata-intensive applications with rondo. *J. Web Sem.*, 1(1):47–74, 2003.

[20] R. J. Miller, M. A. Hernández, L. M. Haas, L. Yan, C. T. H. Ho, R. Fagin, and L. Popa. The clio project: managing heterogeneity. *SIGMOD Rec.*, 30(1):78–83, 2001.

[21] M. Petropoulos, A. Deutsch, and Y. Papakonstantinou. Clide: interactive query formulation for service oriented architectures. In Chan et al. [7], pages 1119–1121.

[22] E. A. Rundensteiner, A. J. Lee, and A. Nica. On preserving views in evolving environments. In F. Baader, M. A. Jeusfeld, and W. Nutt, editors, *KRDB*, volume 8 of *CEUR Workshop Proceedings*, pages 13.1–13.11. CEUR-WS.org, 1997.