VISIONARY: a Visual Query Language Based on the User Viewpoint Approach

Francesca Benzi

C.d.L. in Scienze dell'Informazione, Università di Bologna
Cesena, Italy

Dario Maio

DEIS, Università di Bologna
Bologna, Italy

Stefano Rizzi

DEIS, Università di Bologna
Bologna, Italy

Abstract

Query languages for RDBMSs lack in intuitive understanding; this precludes inexperienced users from taking advantage of them. The adoption of a visual interface can simplify the query formulation process by enabling users to interact with a friendly environment. In this work we propose a visual query language based on a hybrid iconic-diagrammatic paradigm, used for both data and query representation. The external data model is called vision and comprises the visual primitives of concept - represented by the combination of text and icon, and association - represented by a name and a multiplicity for each possible orientation. The external query model is based on the definition of a user viewpoint, which is a perspective for accessing data defined dynamically by selecting a concept of primary interest for formulating each query.

1 Introduction

The fast and continuous evolution of the computer environment, progress in hardware and software and the widespread use of automation in both homes and offices have considerably altered software demand. Not many years ago efficiency was the developers' main target; now, since consumer electronics is the primary growth area for computers, usability is also critical for commercial software. As a matter of fact, there is substantial empirical evidence that attention to usability dramatically decreases costs and increases productivity [8]. On the other hand, users are becoming harder and harder to please: they do not want to waste time reading manuals and they expect to learn to use new products quickly and effortlessly.

Visual design for user interfaces is aimed at improving their usability. In this direction the graphical presentation of the information is essential, since an effective drawing often conveys a concept more immediately and more clearly than a long written explanation [9]. Good usability results have been achieved by applications such as word processors, drawing tools and, in general, by those applications which do not require specific theoretical knowledge to be used.

Unfortunately, query languages for database management systems (DBMSs) lack in both intuitive understanding and visual aid; this precludes inexperienced users from taking full advantage of them. In fact, an inexperienced user willing to query a DBMS by means of a traditional query language must face a number of problems [5]:

- *Learning*: mastering even the basic principles of the relational theory takes a lot of time and study for a novice, and most people do not have such strong determination.
- *Difficult use*: a user querying a DBMS is subjected to rigid syntax, that is hard to remember and with very little visual aid.
- *Poor feedback*: the user receives little feedback about the semantics of the query (s)he is formulating.
- *Little interaction*: the query-building process is usually a command-based mechanism which does not contemplate any form of dialogue with the user.

A visual interface can effectively support the query design process: although the difficulty of the subject cannot be completely overcome, users interact with a friendly and helpful environment, and query formulation becomes easier. Some RDBMSs are provided with visual languages, which generally support a graphical representation of the tables belonging to the database (in most cases a form) and a visual process for join definition. These facilities enable the user to formulate queries without writing SQL instructions, but demand a good knowledge

of relational theory when a new query is to be built. In particular, joins are often formulated by dragging an attribute of the first table onto an attribute of the other table: though join conditions need not be explicitly written, their meaning must be well understood by the user.

According to [2], four visual paradigms can be followed in building a visual interface for a DBMS: tabular, diagrammatic, iconic and hybrid. Among tabular approaches we recall QBE [11], the first successful attempt to build a graphical user interface for an RDBMS, whose basic principles are still used by many commercial products. A diagrammatic approach is followed in GUIDE [10] and in SUPER [7], where queries can be formulated by acting directly on the conceptual scheme, and in QBD* [1], where the user builds a graph representing the query. Some diagrammatic languages adopt a 3-D representation for data and results (see AMAZE [5]). In the iconic approaches, queries are formulated by selecting and combining icons, and by creating new ones. A 3-D iconic approach can be found in [6], where virtual reality techniques are used to query pictorial databases.

The iconic paradigm is very intuitive and has great expressive power. Nevertheless, the capability of distinguishing one icon from the other decreases with the increase of the total number of icons; besides, icons which represent actions or abstract concepts are very difficult to design. Hybrid paradigms attempt to overcome this problem by joining the immediacy of icons with the expressive power of text.

In this paper we propose a visual query language called VISIONARY, based on a hybrid iconic-diagrammatic paradigm used for both data and query representation. In particular, the external data model is called *vision* and comprises the visual primitives of *concept* (represented by the combination of text and icon) and *association* (represented by a name and a multiplicity for each possible orientation); the external query model is based on the definition of a *user viewpoint*, that is, a dynamic perspective for accessing data. The internal data model is relational, and the internal query model is SQL.

VISIONARY is mainly directed to occasional users, who only have a general idea about the contents of the database and are willing to formulate simple queries (respectively, *naive* and *uninformed* users according to the classification reported in [2]). As for most visual query languages, the expressive power of VISIONARY is limited to first order queries (see the enriched version of Chandra's hierarchy of queries in [2]).

Query formulation in VISIONARY shares some ideas with QBD*: in fact, it is carried out by selecting nodes and arcs from a graph-like representation of the database. QBD* enables formulation of recursive queries, which at present cannot be formulated with VISIONARY. On the other hand, the selection of a subscheme and the choice of the paths of joins in VISIONARY can be made in most cases implicitly by selecting a user viewpoint.

Section 2 introduces the working example which will be used throughout the paper. In section 3 we briefly outline the user viewpoint approach, on which the external query model is based. In section 4 we describe our visual query language, and in section 5 we give some examples of query formulation.

2 The "Conference Database"

The conference database models the organization of ACM conferences. Its conceptual model is sketched in Figure 1 using the E/R formalism. A possible logical scheme is represented below (primary keys are underlined; foreign keys are followed by ":" and by the relation scheme they refer to):

```
AREA(codArea, description)
TOPIC(codTopic, description)
INCLUDES(codArea:AREA, codTopic:TOPIC)
WORKGROUP(codGroup, interestedIn:AREA)
CONFERENCE(codConf, title, place, date, about:AREA, organizedBy:WORKGROUP)
PAPER(codPaper, title, #pages, about:TOPIC, sentTo:CONFERENCE, receiptDate)
INVITED_PAPER(codPaper:PAPER, writtenBy:WORKGROUP)
SUBMITTED_PAPER(codPaper:PAPER, accepted?)
RESEARCHER(codRes, name, address, acmMember?, belongsTo:WORKGROUP)
ATTENDS(codRes:RESEARCHER, codConf:CONFERENCE, feePaid?)
CHAIRS(codRes:RESEARCHER, codConf:CONFERENCE, role)
INTERESTED(codRes:RESEARCHER, codTopic:TOPIC)
WRITES(codRes:RESEARCHER, codPaper:SUBMITTED_PAPER, mainAuthor?)
REFEREES(referee:RESEARCHER, codPaper:SUBMITTED_PAPER, evaluation)
```

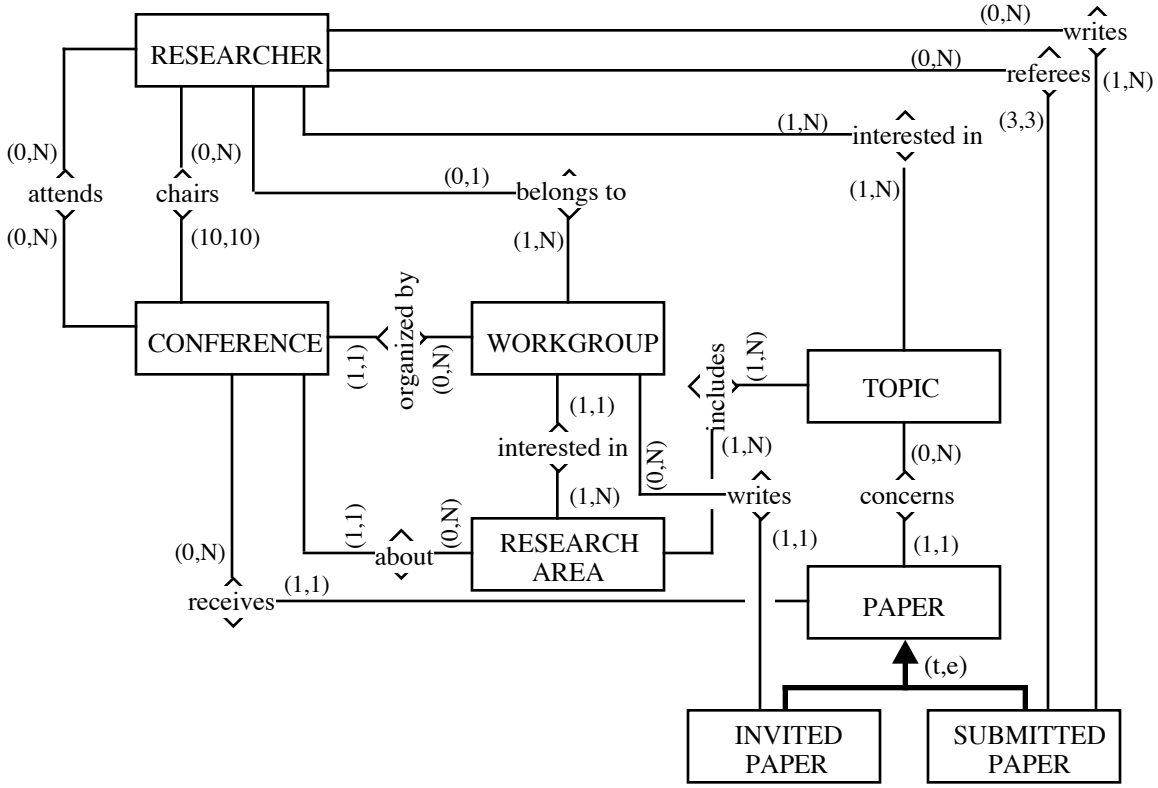

Figure 1: E/R conceptual model of the conference database

3 The User Viewpoint Approach

When conceiving a query on a database, the user of an RDBMS has in mind a set of attributes, belonging to one or more relations, whose values (s)he is interested in obtaining. Among these relations, one has a primary role since it embodies the point of view from which the user wants to access data in the database. For instance, suppose the user querying a university database is interested in knowing, for each teacher, the students (if any) who attend a course held by that teacher. Since the courses *of* a teacher and the students *of* the courses *of* that teacher are required, the primary role is played by the entity teacher; in other words, the attributes of the courses can be considered as "extended" attributes of the teacher, and those of the students as "extended" attributes of the teacher's courses.

In our approach to query inference the user, when formulating a query, can specify a *primary relation* (PR). Each PR defines a different *user viewpoint* (UV), i.e., a different perspective for accessing data; within a UV, the PR is connected to each other relation scheme through exactly one path of logical associations. Thus, within a query interpreted according to the UV defined by its PR, it is possible to reference attributes belonging to any relation scheme without explicitly formulating the necessary joins.

An outline of the UV approach will be given in sections 3.1 through 3.3; a detailed description can be found in [3].

3.1 The Database Graph

Let $\mathcal{D}$ be a database scheme including a set of relation schemes $\mathcal{R}=\{R_i, i=1,...,n\}$ and a set of integrity constraints. Each referential integrity constraint determines a *potential link* (PL) between the attributes involved (i.e., an attribute A_i, which is foreign key in R_i and refers to the primary key B_j of R_j, determines a PL $\{R_i.A_i,R_j.B_j\}$); other PLs may be declared explicitly by the database designer when the comparison between two attributes is relevant. We represent $\mathcal{D}$ by a non-directed graph, called *database graph*, where each node corresponds to a relation scheme R_i and each edge to a PL.

The graph representing the conference database is shown in Figure 2.

3.2 The User Viewpoint

The UV associated to a PR is represented by a spanning directed tree on the database graph, rooted in the PR.

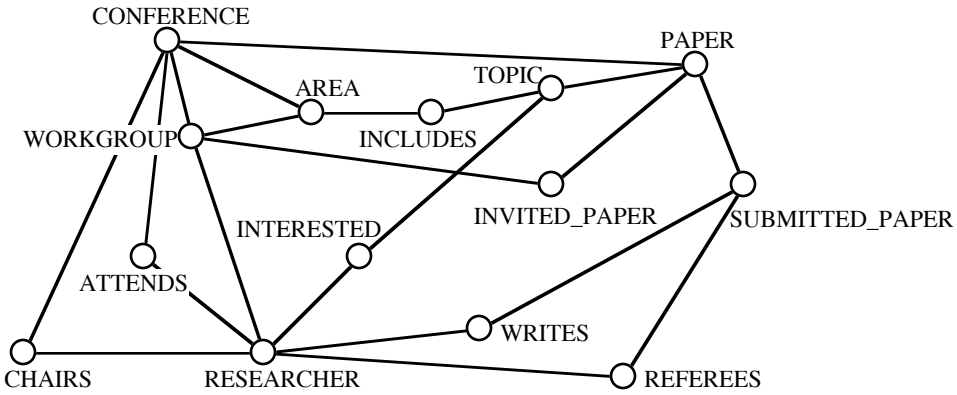

Figure 2: Database graph for the conference database

If the database graph is acyclic, exactly one spanning tree exists for each PR; it can be obtained by giving each PL in the database graph a direction in such a way that no PL enters the PR and all other nodes are entered by exactly one PL. In this case the UV is univocally determined, and only one interpretation is possible for each query sentence.

If the database graph is cyclic, instead, several different spanning trees may correspond to each PR; in order to give exactly one interpretation of each query sentence, a criterion must be used to select, for each PR, one spanning tree (the UV) so that the resulting interpretations are the most reasonable, that is, those which the user most probably expects. In order to select the UV, we estimate the *soundness* of the possible query interpretations by considering the properties of the PLs involved; the UV associated to a PR is then defined as the spanning tree which maximizes the soundness for the queries formulated from that PR. The three PL properties taken into account for estimating the soundness are the strength of the PL (explicitly defined by the designer or implicitly determined by the names of the attributes involved in the PL), multiplicity (-to-one or -to-many) and frequency (how many times the PL has been employed by the user for query formulation).

Figure 3 shows the UV associated to PR PAPER.

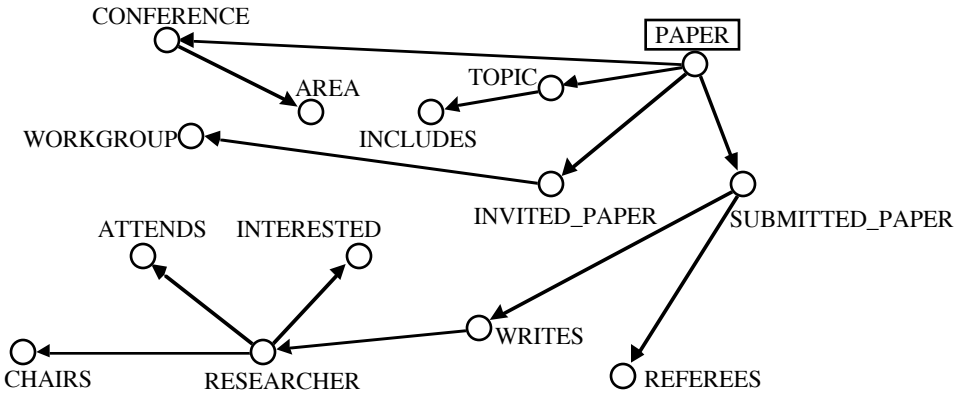

Figure 3: UV associated to PR PAPER

3.3 Query Interpretation

Query interpretation is carried out within the UV associated to the PR used for the query. Each query is translated into an unambiguous representation by generating an equi-join for every PL belonging to the directed paths which, within the UV, connect the PR to every other relation scheme mentioned in the query sentence (within a spanning tree, the root is connected with each other node through exactly one directed path).

For instance, consider a query having PR PAPER and mentioning attributes *PAPER.title*, *TOPIC.description*, *AREA.description*, *RESEARCHER.name*. The interpretation adopted for this query can be expressed in SQL as follows:

```
select   PAPER.title, TOPIC.description, AREA.description, RESEARCHER.name
from     PAPER, TOPIC, CONFERENCE, AREA, WRITES, RESEARCHER
where    PAPER.about = TOPIC.codTopic
and      PAPER.sentTo = CONFERENCE.codConf
and      CONFERENCE.about = AREA.codArea
and      PAPER.codPaper = WRITES.codPaper
and      WRITES.codRes = RESEARCHER.codRes
```

Consistently with the UV shown in Figure 3, for each paper we interpret TOPIC as the topic of the paper, AREA as the theme of the conference the paper was sent to, and RESEARCHER as the researcher who wrote the paper.

4 VISIONARY

VISIONARY is a visual language which enables an inexperienced user to access a relational database through an effective iconic-diagrammatic representation and to formulate queries intuitively without knowing the details of the relational theory. Section 4.1 proposes the visual metaphor used for representing the database; section 4.2 describes the query language, while section 4.3 discusses how queries are translated into SQL.

4.1 Visions

A *vision* is an abstract representation of a database, built by an expert user taking into account both the semantics of the data scheme and the needs of the inexperienced users who will access the database through the vision itself. Several visions may be built on the same database, aimed at addressing different classes of users and/or different application requirements.

Let $\mathcal{D}$ be a database scheme, $\mathcal{R}=\{R_i, i=1,...,n\}$ the set of its relation schemes, and $\mathcal{A}_i$ the set of attributes of R_i. A vision on $\mathcal{D}$ consists of a set of visual objects, which may be *concepts* or *associations*.

A concept corresponds to exactly one relation scheme $R_i \in \mathcal{R}$, and is defined by a name, an icon and a set of attributes belonging to $\mathcal{A}_i$. Figure 4 shows an example of how a concept is represented visually; the attributes of the concept may be viewed by double-clicking the icon.

Figure 4: Visual representation of a concept

An association expresses the relationship between two or more concepts. An association between two concepts corresponding to R_i and R_j may correspond either to one PL $\{R_i.A,R_j.B\}$ or to two PLs $\{R_i.A,R_Z.T\},\{R_j.B,R_Z.U\}$, where R_Z does not correspond to any concept (*ghost scheme*). The second form is generally used when R_Z is the relation scheme modelling the n:m relationship between R_i and R_j, and is not considered worth being represented as a concept. An association between three concepts corresponding to R_i, R_j and R_k corresponds to three PLs $\{R_i.A,R_Z.T\}$, $\{R_j.B,R_Z.U\}$, $\{R_k.C,R_Z.V\}$, where R_Z is the ghost scheme and models the ternary relationship; similarly for associations between four or more concepts.

An association between m concepts is defined by m names, m minimum cardinalities, m maximum cardinalities and by a set of attributes belonging to the ghost scheme (if any) or to the relation schemes corresponding to the concepts involved. The i-th name is that given to the association when "read" starting from the i-th concept; the i-th minimum and maximum cardinalities express the number of instances of the other m-1 concepts associated to one instance of the i-th concept (min. cardinality is 0 or 1; max. cardinality is 1 or N).

In the conference database, an example of binary association without a ghost scheme is the following:

```
(  PLs: {{PAPER.sentTo,CONFERENCE.codConf}}
   start: PAPER   ( name: is sent to
                    card: 1-1   )
   start: CONFERENCE   ( name: receives
                         card: 0-N   )
   attributes: {CONFERENCE.receiptDate}    )
```

An example of binary association with a ghost scheme (*WRITES*) is:

```
(  PLs: { {RESEARCHER.codRes,WRITES.codRes},
          {SUBMITTED_PAPER.codPaper,WRITES.codPaper} }
   start: RESEARCHER   ( name: writes
                         card: 0-N   )
   start: SUBMITTED_PAPER   ( name: is written by
                              card: 1-N   )
   attributes: {WRITES.mainAuthor?}    )
```

Figure 5 shows an example of how the multiplicity of an association may be visualized by clicking on one of the concepts involved.

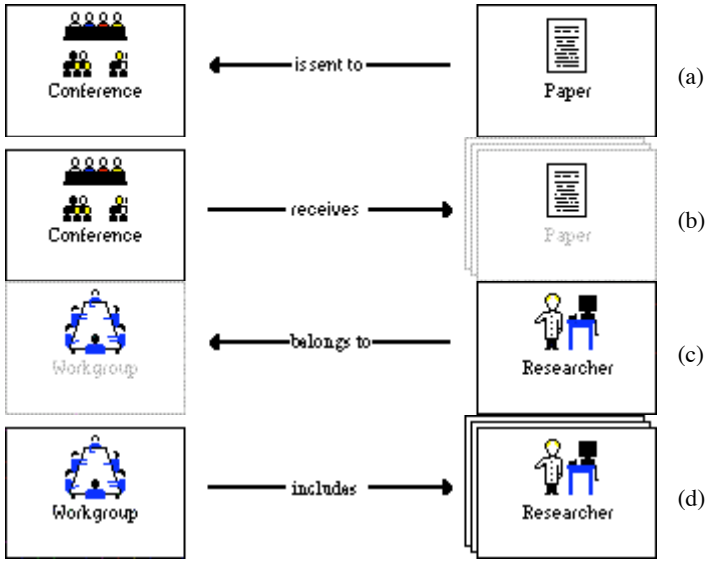

Figure 5: Visual representation of associations with cardinality 1-1 (a), 0-N (b), 0-1 (c), 1-N (d)

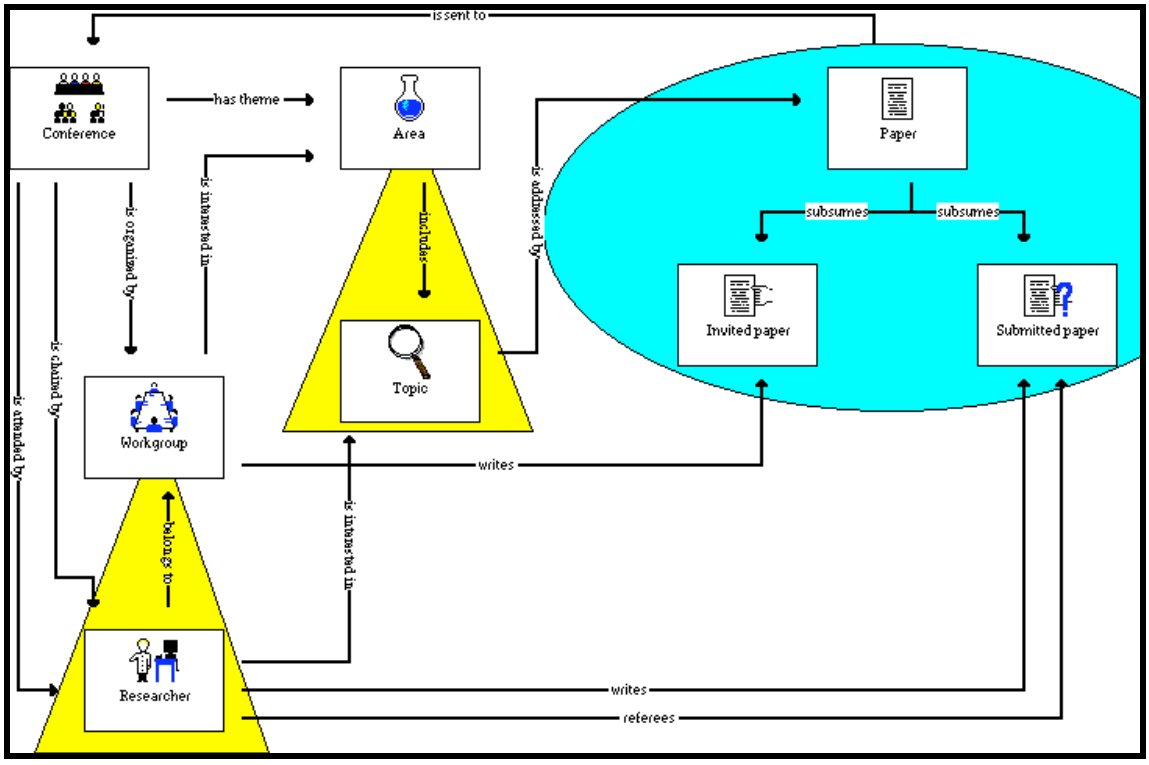

Figure 6: A vision on the conference database

The visual objects and their properties, as well as the graphical layout of the vision, are completely defined by the expert user. Please note that not every relation scheme needs be represented by a visual object, nor does every attribute of a relation scheme represented by a visual object need to be included in the object itself; thus, a vision may be used to hide a part of the database from the inexperienced user.

An example of a complete vision on the conference database is shown in Figure 6. Ellipses and triangles are part of the visual representation of associations with semantics IS-A and PART-OF, respectively.

4.2 Visual Formulation of Queries

Inexperienced users access the database through the visions previously defined by expert users. The intuitiveness and expressivity of visions help inexperienced users to understand the database semantics, and assist them in formulating queries.

The formulation of a query on a vision consists of five steps (steps 2., 4., and 5. are optional):

1. Choose a viewpoint (implicit formulation of joins).
2. Edit the vision (explicit formulation of joins).
3. Choose attributes to be retrieved (projection).
4. Formulate restrictions on concept attributes (selection).
5. Order and/or group the results.

The first step is executed visually by clicking on a concept, whose corresponding relation scheme becomes the PR. The vision is modified by disabling (painting grey) some associations and by orientating the others according to the UV associated to the PR. An association is disabled if one or more of the corresponding PLs are not included in the UV. The enabled associations determine the interpretation given to each concept in the vision. Consider for instance the UV in Figure 3, associated to PR PAPER; the vision from viewpoint *Paper* is shown in Figure 7. The association between *Researcher* and *Conference* is disabled since the PL {CONFERENCE.codConf, ATTENDS.codConf} is not part of the UV. The interpretations given to concepts *Conference* and *Researcher* are, respectively, the conference the paper has been sent to, and the researcher who wrote the (submitted) paper. Figure 8 shows the vision from viewpoint *Researcher*.

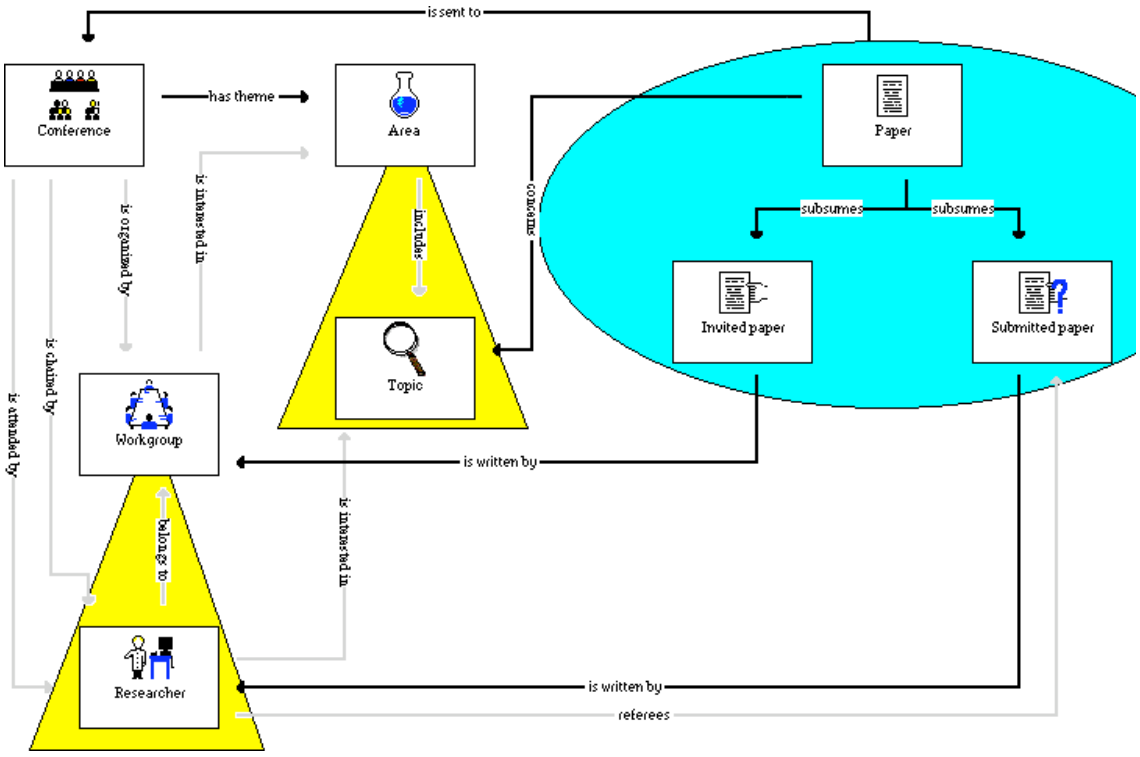

Figure 7: Vision on the conference database from viewpoint *Paper* (disabled associations in grey)

The second step may be executed if the enabled associations proposed by the system are not those the user is interested in. Suppose that, for instance, the user wants to know, for each paper, the area including the topic of the paper. In the interpretation proposed by the system from viewpoint *Paper*, *Area* is the theme of the conference the

paper has been sent to; thus, the user must disable the association *has theme* and restore the association *belongs to* from *Topic* to *Area*. This action changes the interpretation given to queries by forcing the formulation of paths of joins different from those provided by the original UV.

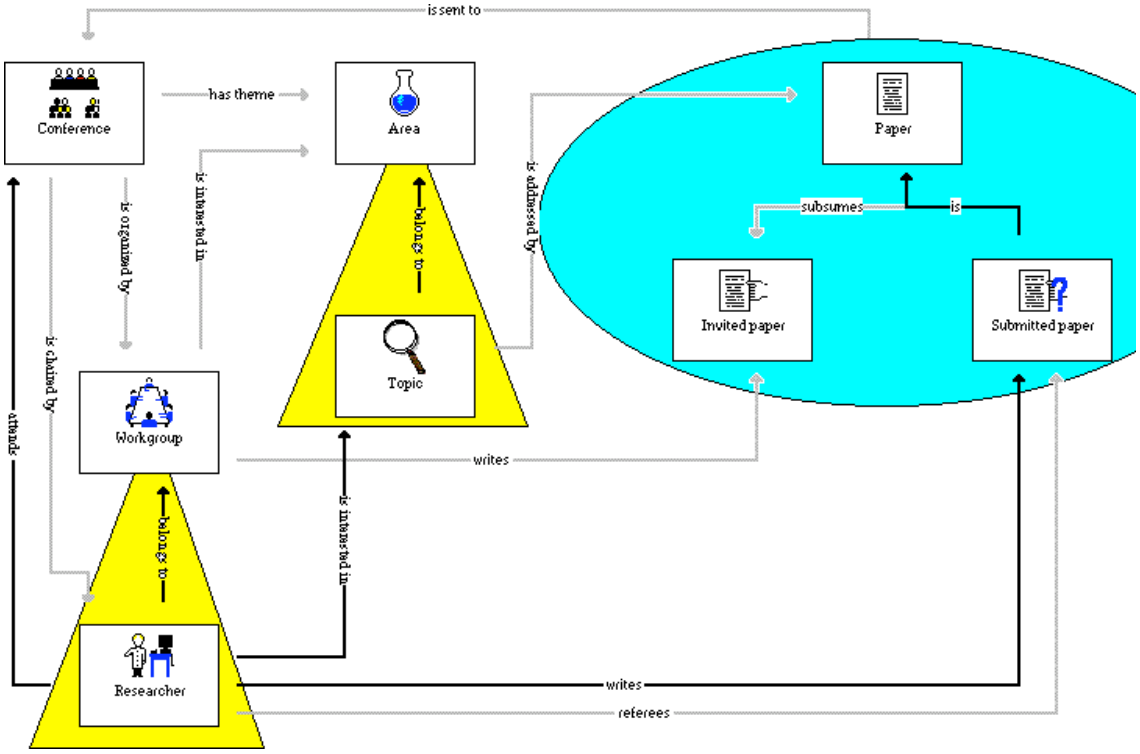

Figure 8: Vision on the conference database from viewpoint *Researcher*

Editing the vision may lead to an attempt to form a cycle; a cycle occurs when two associations enter a given concept. In this case the concept is doubled within the vision, both graphically and logically. Suppose the user wants to retrieve, for each researcher, the conferences attended and the conferences chaired. If, from viewpoint *Researcher*, association *chairs* is restored without disabling association *attends*, the concept *Conference* is doubled. Thus, the user may access contemporarily both interpretations of concept *Conference* (see Figure 9.a). If the user is interested in the themes of conferences, (s)he can click on association *has theme*. The association is graphically doubled, and the user can choose to retrieve either the areas of the conferences attended, or the areas of the conferences chaired (see Figure 9.b), or both; in the latter case, concept *Area* is doubled. Another example of concept doubling can be found in section 5 (example #3). It should be noted that, owing to the doubling of concepts, the enabled associations never form cycles.

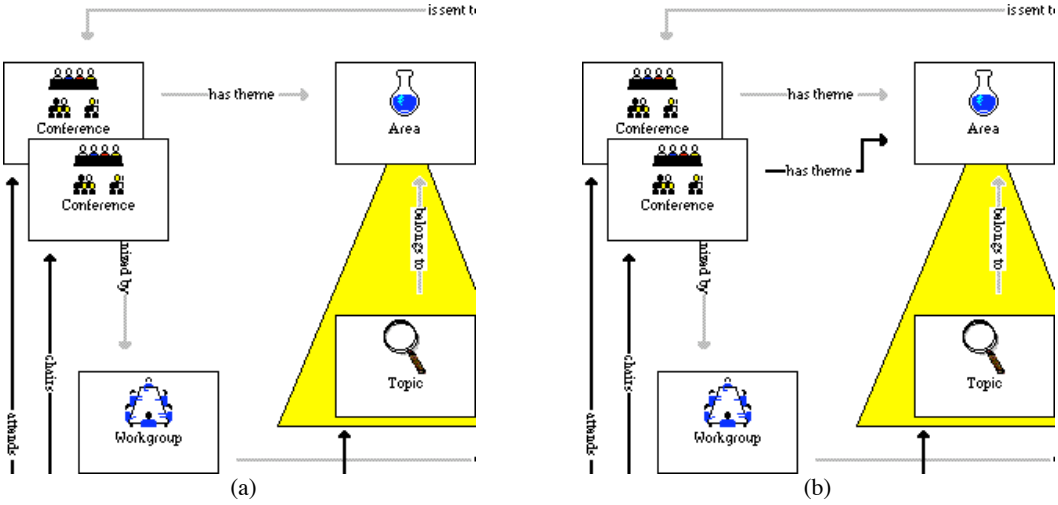

Figure 9: (a) Doubling concept *Conference*; (b) Restoring association *has theme*

The third step is carried out by double-clicking on concepts and selecting the attributes to be displayed in the result.

At the fourth step, a selection condition can be formulated by double-clicking on a concept and writing a predicate. A predicate consists of an attribute (picked from the list of attributes of the concept), an operator (picked from a list including all the standard SQL operators) and one or more values (typed by the user).

The last step may be performed on a window showing a preview of the results in tabular form. When choosing to order or group data according to a specific attribute, the user will see the results of his/her action on a sample set of data.

Query formulation may be carried out in "preview mode": in this case the inexperienced user can see, at each step, the results of the query (s)he is formulating and verify its correctness. Since execution of a complex query on a large database may take a long time, preview mode can be disabled in order to avoid slowing down the formulation phase too much.

4.3 Query Interpretation

In this section we show how a visual query is interpreted and translated into an SQL query.

We say a concept or an association is mentioned in a visual query q if at least one of its attributes has been chosen to be retrieved or is involved in a selection predicate. On the database graph, we say a relation scheme is mentioned in q either if it corresponds to a concept mentioned in q or if it is the ghost scheme of an association mentioned in q.

Query interpretation is based on the associations which are enabled within the vision at the time the query is executed. If, when formulating q, the user has left the vision unchanged, the SQL interpretation is built as shown in section 3.3, based on the UV with PR corresponding to the concept used as viewpoint.

On the other hand, if the vision has been edited, the UV must be modified accordingly. For each association that has been disabled, all the corresponding PLs are dropped from the UV; for each association that has been restored, all the corresponding PLs are inserted into the UV. If a concept has been doubled (meaning that different interpretations of that concept are considered contemporarily), the corresponding relation scheme in the UV is also doubled, and the two copies are discriminated by using aliases (thus, the modified UV never has cycles). If, within the modified UV, there is a path of PLs connecting the PR with each relation scheme mentioned in q, the SQL interpretation is still built as shown in section 3.3, based on the modified UV. Otherwise, the modified UV is non-connected: an equi-join is generated for every PL belonging to the paths which, within each connected sub-tree, connect the root to each other relation scheme mentioned; from a conceptual point of view, since no joins connecting the different sub-trees are provided, the query returns the Cartesian product between the interpretations given to the connected portions of the vision.

The attributes selected by the user to be retrieved are inserted in the select list of the SQL query; the selection predicates formulated visually are inserted into the where clause. Groupings and/or orderings imposed on data are translated using the corresponding SQL clause.

Once the visual query has been translated and executed, its results are shown to the user in tabular form.

5 Examples

1. *For each paper sent to conference IDS after January 1st 1995, retrieve its title and its topic. The results must be ordered alphabetically by title.*

This query is formulated visually by choosing *Paper* as the viewpoint; the vision needs not be edited. The projection is carried out by double-clicking on *Paper* and *Topic*, and selecting attributes *title* and *description*, respectively. The selection is carried out by double-clicking on *Conference* where the predicate *title equal "IDS"* is formulated, and on association *is sent to* where the predicate *receipt date greater than 1/1/1995* is formulated. On the preview window, the ascending ordering on *title* is imposed.

2. *Retrieve the title of the papers submitted by researcher Smith, together with the title and date of the conferences the papers were sent to.*

This query is formulated visually by choosing *Researcher* as the viewpoint. According to the vision from viewpoint *Researcher* (see Figure 8), the interpretation given to concept *Conference* is the conferences attended by a researcher; since this is not the interpretation required, the user must edit the vision by disabling the association *attends* and restoring the association *is sent to*. The projection is carried out by choosing attributes *titles* from *Paper* and *title*, *date* from *Conference*; the selection is carried out by formulating the predicate *name equal "Smith"* on *Researcher*.

3. *For each paper sent to a conference with theme "Artificial Intelligence", retrieve the title of the paper and the area including the topic of the paper.*

This query is formulated visually by choosing *Paper* as the viewpoint. The concept *Area* is used twice, with different interpretations: the area of the conference to which the paper was sent (this interpretation is the one implicitly given within the vision from viewpoint *Paper*), and the area including the topic of the paper. If association *belongs to* between *Topic* and *Area* is restored (without disabling association *has theme*), the concept *Area* is doubled. Thus, the projection is carried out by choosing attribute *description* from *Area* (the one associated to *Topic*); the selection is carried out by formulating the predicate *description equal "AI"* on *Area* (the one associated to *Conference*).

6 Conclusions

In this work we have described a visual query language based on a hybrid iconic-diagrammatic paradigm. The user perceives the database through a metaphor called vision, which comprises concepts and associations. Queries are formulated by choosing a viewpoint, deciding the attributes to be retrieved and expressing selection predicates. If the interpretation given to a query is not the one the user had in mind, the user can force a different interpretation by disabling some associations and restoring others.

The language is still at a prototypical stage. Currently we are working on improving the visual metaphor which models the database and on extending the expressive power of the visual language, in order to enable users to formulate even complex queries in a completely visual fashion. As to the first issue, we will investigate the possibility of giving an effective visual representation to attributes (currently, the attributes of a visual object are listed, in textual form, by clicking on the object itself) and to retrieved data. As to expressive power, in [4] we have shown an SQL extension where multiple viewpoints could be adopted for query formulation; in VISIONARY, the user selecting two or more viewpoints within a single query will be enabled to choose between as many different interpretations for the concepts included in the vision.

7 References

[1] Angelaccio M, Catarci T, Santucci G. QBD*: a graphical query language with Recursion. IEEE Trans Software Engineering 1990; 16(10):1150-1163.

[2] Batini C, Catarci T, Costabile MF, Levialdi S. Visual query systems: a taxonomy. In: Visual Database System II. Elsevier Science Publishers B.V, North-Holland, 1992, pp.153-168.

[3] Bellavia G, Maio D, Rizzi S. Minimizing the cost of query formulation through User Viewpoint Relations. In: Proc Secondo Convegno Nazionale su Sistemi Evoluti Per Basi Di Dati. Rimini, Italy, 1994, pp. 141-159.

[4] Bellavia G, Maio D, Rizzi S. An SQL extension supporting user viewpoints. In: Proc 6th Int Conf on Database and Expert Systems Applications. London, 1995, pp. 334-343.

[5] Boyle J, Leishman S, Fothergill J, Gray P. Design of a Visual Language for a Database. Technical Report, University of Aberdeen, 1994.

[6] Del Bimbo A, Campanei M, Nesi P. A 3D Visual Environment for Querying Image Databases. In: Proc Intern Workshop of Advanced Visual Interfaces. World Scientific Publishing Co. Ltd., Singapore, 1992, pp. 12-25.

[7] Dennebouy Y, Andersson M, Auddino A et al. Super: Visual Interfaces for Object + Relationship Data Models. Journ of Visual Languages and Computing 1995; 5:73-99.

[8] Myers BA. Why are Human-Computer Interfaces Difficult to Design and Implement?. Technical Report CMU-CS-93-183, Carnegie Mellon University, School of Computer Science, 1993.

[9] Tufte E. The visual display of quantitative information. Graphic Press, 1983.

[10] Wong HKT, Kuo I. GUIDE: Graphical User Interface for Database Exploration. In: Proc 8th VLDB Conference. Mexico City, 1982, pp.22-31.

[11] Zloof MM. Query-by-Example. In: Proc AFIPS Conf, National Computer Conference, vol. 44, 1975, pp.431-438.