ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Jessie Kennedy and Peter Barclay (Eds)

Interfaces to Databases (IDS-3)

Proceedings of the 3rd International Workshop on Interfaces to
Databases, Napier University, Edinburgh, 8-10 July 1996

Paper:

VR Queries and Their Transformations in a Progressive Querying Environment

S.K. Chang, M.F. Costabile and M. Vairo

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

VR Queries and their Transformations in a Progressive Querying Environment

S. K. Chang

Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
Phone +1-412-6248423, chang@cs.pitt.edu

M. F. Costabile

Dipartimento di Informatica, Università di Bari, Via Orabona 4, 70126 Bari, Italy
Phone +39-80-5443300, fcosta@iesi.ba.cnr.it

M. Vairo

Dipartimento di Informatica, Università di Bari, Via Orabona 4, 70126 Bari, Italy
Phone +39-80-5443300, fcosta@iesi.ba.cnr.it

Abstract

We have proposed a progressive querying environment that enables the user to interact with the database by a sequence of partial queries, and to visualize both database queries and retrieval results in many different representations. In our approach, the user can query the database in the Virtual Reality (VR) space, and then switch to the logical space to continue the formulation of the query. This paper provides a formal definition and characterization of VR queries, describes what database operators are applicable under the different interaction paradigms, and investigates how VR queries can be transformed into other types of queries, so that they can be utilized as part of a progressive querying system.

1 Introduction

Virtual Reality (VR) techniques are introduced as a new interaction paradigm particularly suited to novice and/or occasional users. Such techniques combine the advantages of 3D visualizations with the power of metaphorical representations. The users are facilitated in the database navigation since the system proposes them an environment that reproduces a real situation and gives the possibility of interacting by manipulating objects that have a direct correspondence with known objects. Moreover, VR scenarios are enriched with many details, taken from real contexts, so that users can easily locate the objects of interest; they can easily understand where they are, so reducing the risk of getting lost in an unknown system.

Current research exploits VR in the visualization of large amount of data (the extension of the database), or in the visualization of complex data objects and their relationships (the intension of the database). In [7] we have proposed an environment that enables the database users to visualize both database queries and retrieval results in many different representations. By exploiting the concept of progressive querying, the user interacts with the database by a sequence of partial queries. In our approach, the user can query the database in the VR space, and then switch to the logical space to continue the formulation of the query. With the incorporation of spatial operators, we can formulate VR query, for example, to find those hospital beds that are located in the ward near a selected bed and occupied by male patients. Such VR query is more powerful than browsing. Therefore, VR is regarded as another visual interaction paradigm, and integrated into both the querying mechanism and the visualization of query results.

However, VR queries may have certain limitations because it may be difficult to express abstract relationships in VR queries. The purpose of this research is first of all to provide a formal definition and characterization of VR queries, secondly to study what database operators are applicable under the different interaction paradigms, and thirdly to investigate how VR queries can be transformed into other types of queries, so that they can be utilized as part of a progressive querying system. A graph query model serves as the underlying unified model, so that different types of queries can be first transformed into this graph query model and then transformed into the target query language [3, 6]. When the user switches from one querying paradigm to another querying paradigm, the (partially formulated) query can also be transformed from the original paradigm to the new paradigm. Therefore, our approach not only allows the user to switch modes, but also the progressive formulation

of a query in several different paradigms. The admissibility condition specifies whether a query can be transformed into another paradigm. The underlying graph query model provides a unified framework to precisely characterize such transformations [6].

This paper is organized as follows. In Section 2, we present the concept of logical information space and physical information space. The definition of VR query is given in Section 3. The database operators that are available in various interaction paradigms are compared and categorized in Section 4. In Section 5, the expressive power of the visual paradigms is discussed. Section 6 discusses the combination of VR queries with other paradigms in a progressive querying system. Section 7 gives the formal characterization of admissible VR queries, which are the sub-queries that can be transformed into queries in other paradigms. Finally, conclusions are drawn in Section 8.

2 Information Spaces

In a visual interface for databases, the information stored in the database needs to be visualized in an information space. This visualization can either be carried out by the user in the user's mind, in which case it is essentially the user's conceptualization of the database; or the visualization could be accomplished by the system, in which case the visualization is generated on the display screen. In this section, we describe the different representations of the information space.

Database objects, in general, are abstracted from real-life objects in the real world. Therefore, we can distinguish the logical information space and the physical information space. In the logical space, the abstract database objects are represented. In the physical space, the abstract database objects are materialized and represented as physical objects such as images, animation, video, voice, etc. The physical objects either mimic real-life objects such as objects in a virtual reality, or reflect real-life objects such as diagrams, icons and sketches.

The real world, from which the database objects are abstracted, is the environment that the database objects must relate to. The real world also is often abstracted in the information space. Only in the virtual reality information space will the real world be represented in a direct way.

The logical information space is a multi-dimensional space, where each point represents an object (a record, a tuple, etc.) from the database. A database object e_j (or an example) is a point in this space. Conceptually, the entire information space then corresponds to all the database objects in a database. The logical information space is thus a unified view of the database, i.e. a universal relation. Each attribute of a database object represents one dimension in this multi-dimensional space. Therefore, in the logical information space, different dimensions actually have different characteristics: continuous, numerical, discrete, or logical.

A query q_i is an arbitrary region in this information space. A clue x_k is also an arbitrary region in the logical information space, but it may contain additional directional information to indicate visual momentum, i.e. the preferred direction of navigation or search. Therefore, an example e_j is a clue. A visual query q_i is also a clue.

The information retrieval problem is to construct the "most desirable" query q_i with respect to the examples e_j and the clues x_k presented by the user. The "most desirable" query is one which will retrieve the largest number of relevant database objects and whose "size" in the information space is relatively small (precision and recall, as defined in literature on information retrieval). The process of visual reasoning may help the user find the most desirable query from examples and clues.

The logical information space may be further structured into a logical information hyperspace, where the clue becomes hyperlinks that provides directional information, and the information space can be navigated by the user by following the directional clues. Information is "chunked", and each chunk is illustrated by an example (the hypernode).

The physical information space consists of the materialized database objects. The simplest example is as follows. Each object is materialized as an icon, and the physical information space consists of a collection of icons. These icons can be arranged spatially, so that the spatial locations approximately reflect the relations among database objects. More recently, intelligent visualization systems are being developed, for task-specific visualization assistance [12]. Such systems can offer assistance in deriving perceptually effective materialization of database objects.

In the physical information space, the objects reflect real-world objects, but the world is still an abstract world. One further step is to present information in a VR information space. VR allows the users to be placed in a 3D environment they can directly manipulate. What the users see on the screen will be the same as what can be experienced in the real world. 3D features can be used to present the results in a VR setting. For example, the physical location of medical records can be indicated in a (simplified) 3D presentation of a Virtual Medical Laboratory by blinking icons. If the database refers to the books of a library, we can represent a Virtual Library in which the physical locations of books are indicated by blinking icons in a 3D presentation of the book stacks of the

library. What the user sees on the screen will be the same (after simplifications) as what can be experienced in the real world. VR such as the Virtual Library or the Virtual Medical Laboratory can become a new query paradigm [7]. For example, the user can select a book by picking it from the shelf, like in the real world.

It is worth noting that we are talking about "nonimmersive" VR [16] i.e. the user is placed in a 3D environment he or she can directly manipulate without wearing head-mounted stereo displays or special gloves, but acting only with mouse, keyboard, and monitor of a conventional graphics workstation. Nonimmersive VR is a valuable interaction paradigm that will be fruitful in database applications, as well as in general business applications. When easily manageable and not intrusive displays and input devices will be available, immersive VR may become acceptable.

3 Definition of VR system and VR query

The VR information space is very different from the physical information spaces discussed in the previous section, also for its expressive power. In VR, movement operators can be specified, i.e. operators that allow to move into the VR physical space and, therefore, to navigate into the information space or mark a part of the information space. Examples are those specifying spatial operations, such as above, below, left, right, forward, backward, that allow the observer to move into an environment and mark a subspace. Usually, such operators are implemented by means of tools, like a mouse or a joystick, or by head movements if equipped with specific helmet.

Another very useful movement operator for navigating into VR is the *zoom-in* operator, that gives the user the possibility of investigating a selected portion of the world. Different uses of the zoom operator have been proposed in the literature [13, 18]. The most significant distinction to be done is between visual (or graphical) zoom and semantic zoom. The former is the closest to the original meaning of the zoom term, which originates from the typical operation performed by cameras, that allows to reduce the visual field, thus magnifying a portion of the original view in order to display more details about the objects. We have called such an operator *Magnify*, and it is usually available in all types of physical information space. Conversely, semantic zoom involves changing the data objects being displayed; the original image is substituted with another one showing the selected region at a different level of detail. For example, if we are moving in a building and different floors are visualized, a double-click on a specific floor will show on the computer screen a view of that floor. The analogous operator zoom-out is defined, thus allowing to go back to the more general view. In the following, we will limit our discussion to the zoom-in operator, since the zoom-out acts in the reverse way. Moreover, we call such an operator *Zoom-in_Detail*, since other zoom-in operators can be defined working in other physical spaces, as it will be shown in the next section.

The only query operator available in VR is *DM_select*, i.e. with a simple mouse click the user indicates a specific object (DM stands for direct manipulation). This is the basic operator of the direct manipulation interaction paradigm, and it must not be confused with the select operator of the relational algebra.

We can formally define a VR system and a VR query as in the following:

Definition 1: A VR system is a 5-tuple (S, O, G, OP_m, OP_q) where:

- S is the set of visible objects (these are all the objects in the 3D world; note that only some visible objects are also database objects)
- O is the set of database objects
- G is a function from O to S
- OP_m is the set of movement operators
- OP_q is the set of query operators.

If not all database objects are visible, then G is a partial function.

Definition 2: A VR query is of the form $OP_q(s_1, s_2, \dots, s_n)$, $s_i \in G(O)$.

As we said above, the only query operator is *DM_select*; it can be applied to one or more objects of the database that are visible in a scene and inside the marked part of the information space, and it allows retrieving information on the selected object(s). In the next section, we will compare the operators available in VR with those available in other visual paradigms.

In order to provide an example of VR system, let us consider a hospital database. A VR interface should propose an environment that reproduces a real situation and gives the users the possibility of interacting by manipulating objects that have a direct correspondence with known objects. Therefore, for the hospital database we will describe what is seen in the reality when referring to a hospital. At the topmost level, it is a set of different wards. Then, we can detail the situation of each ward, as composed of different floors, where rooms for doctors,

rooms for patients, labs, etc. are located. We can further detail the various rooms and the labs, up to include those objects that have a correspondence with the objects stored into the database, i.e. the objects in $G(O)$.

Figure 1 describes the organization of the virtual hospital. The topmost level represents the scene shown on the computer screen when entering the system. It refers to the hospital as the set of the wards belonging to it; in the example, four wards are included. This VR scene is shown in Figure 2, where each of the four wards is represented as a separate building. The user who wants to explore a ward must zoom-in into it.

Figure 1: Organization of the virtual hospital

As indicated by the legend at the bottom of Figure 1, a VR environment is denoted by an oval. The other node in the schema in Figure 1 is a rectangle, denoting an object, included in a VR environment, that is also a correspondent of a database object; in other words, the rectangle denotes an object in $G(O)$. A VR environment is

also seen as an object in S . By applying the *Zoom-in* operator on an object ward, a detailed view of it is obtained, that refers to the ward reception. From the reception, five more environments can be detailed, namely *lab_test*, patient information, doctor information, male division, female division. *Lab_test* is a VR environment from which information about the tests performed in the labs can be obtained. Patient information and doctor information give the possibility of inquiring about patients and doctors respectively. From the male division environment (and the female division as well) the room environment can be detailed, where data about bed occupation can be obtained. Moreover, in the ward reception there is also the possibility of directly inquire the database through the object *reception desk*, that is an element in $G(O)$, in order to obtain information about the location of patients in the ward. The arc with arrow at both extremes indicates that the two environments it connects are directly reachable one from the other.

Figure 2: Initial scene for the virtual hospital; the four buildings represent four wards

Our formal definitions of both VR and the mapping between database objects and visible objects is very similar to the one given in Virgilio, a system that generates VR-based visualizations of complex data objects [14]. The Virtual World is defined in Virgilio as a set of virtual world objects that can be visualized in a scene. Virtual world objects are partitioned into three groups: *containers*, *accessories*, and *decorative objects*. A container is an object related to other objects according to a containment relation, that can be of two forms: *classification* and *aggregation*. A container is a classifier when it classifies a set of objects of the same type; for example, a file cabinet classifies objects of type folder. If a container includes objects of different type, it is an aggregator; as an example, a desk is an aggregator since it can include an object of type book and an object of type portrait case. An accessory is a support where one or more types of data (such as text, strings, pictures, etc.) can be displayed. Therefore, it provides a metaphor for representing a database object. A decorative object is added in the VR scene to facilitate the perception of the context to the user.

The Virgilio's partition of the virtual world objects is similar to the partition of the set of visible objects S we have defined. The VR environments we have indicated in the schema in Figure 1 are examples of containers that can either be classifiers or aggregators. For instance, the hospital environment, that includes four wards, is a classifier, while the ward reception environment is an aggregator, being composed by non-homogeneous

environments. The set of objects $G(O)$ we have defined are the equivalent of Virgilio's accessories, since they are the correspondent, in the virtual world, of the objects stored in the database. Finally, we also included in the VR scenes decorative objects, i.e. objects that do not map database objects but only contribute to a better rendering of the VR scene. An example are the objects *car* shown in the scene in Figure 2.

4 Database operators in VR and other visual paradigms

Traditionally, the reference for evaluating the expressive power of database query languages has been the relational algebra. The classical operators are select, project, join, plus those performing set operations, namely union, intersection and difference. To better characterize the VR paradigm, in this section we discuss the operators available in VR and compare them with those of the visual paradigms analyzed in [5]. We take into account query operators as well as navigation operators; the latter are essential in visual query systems.

The query paradigm, which settles the way the query is performed and represented, is very much dependent on the way the data in the database are visualized. The basic types of visual representations are form-based, diagram-based, and icon-based, according to the visual formalism primarily employed, namely form, diagram, and icon [5]. Form-based representations have been the first attempt to provide users with friendly interfaces for data manipulation, taking advantage of the bi-dimensionality of the computer screen. The queries are formulated by filling appropriate fields of prototypical tables that are visualized on the screen. Representations based on diagrams are widely adopted in existing VQSs. The word diagram refers to any graphics that encodes information using position and magnitude of geometrical objects and/or shows the relationships among components. Diagrammatic representations adopt as typical query operators the selection of elements, the traversal on adjacent elements and the creation of a bridge among disconnected elements. The icon-based representation uses sets of icons to denote both the objects of the database and the operations to be performed on them. In iconic VQSs, a query is expressed primarily by combining icons.

	<i>DM_Select</i>	<i>RA_Select</i>	<i>Project</i>	<i>Join</i>	<i>Union</i>	<i>Intersec.</i>	<i>Difference</i>	<i>Details</i>
FORM	P	P	P	P	P	P	P	P
DIAGRAM	P	X	P	P	X	X	X	
ICON	P	X	X	X	X	X	X	
VR	P							

(a)

	<i>Spatial Operators</i>	<i>Zoom-in _Detail</i>	<i>Zoom-in _Subset</i>	<i>Zoom-in _Fuzzy</i>	<i>Magnify</i>	<i>Scroll</i>
FORM		X	X	P	V	V
DIAGRAM		X	P		V	V
ICON		X	P		V	V
VR	P	P			V	V

(b)

Table 1: Database operators for the different visual paradigms: (a) query operators; (b) navigation operators. The symbol in a table cell is P if the operator specified in the cell column is available in the visual paradigm indicated in the cell row; it is X if the operator is available only in some cases; it is V if the operator only involves view modification. The cell is empty if the operator is not available in the visual paradigm.

We have summarized the database operators in Table 1. The operators are indicated in the table columns, while the rows specify the four visual paradigms analyzed. Table 1a considers the query operators, namely those defined in the relational algebra, plus two more operators: *DM_Select* defined in the previous section and specified in the first column (by *RA_Select* we indicate the select operator of the relational algebra), and *Detail*, that allows to visualize all the details of an object, for example all its attributes, that are usually listed as a form. Table 1b considers the operators that allow the user to navigate to reach the data of interest that will be visualized on the computer screen. A table cell contains the symbol P if the operator specified in the cell column is available in the visual paradigm indicated in the cell row; the symbol in the cell is X if the operator is available only in some cases. For example, in some icon-based systems, the project operation is implemented by the overlapping of certain icons [17]. The table cell is empty if the operator is not available in the visual paradigm; this is the case of the relational algebra operators for the VR paradigm.

The first column of Table 1b specifies the spatial operators typical of the VR paradigm (see Section 3). They are not available in the other three paradigms. We then consider three different zoom-in operators of semantic type. For the sake of brevity, corresponding zoom-out operators, that behave in the opposite way, are omitted in the table. It is worth noting that zoom operators are proposed in the literature with different meaning. Sometimes, the same operation has been called with different names in different systems. We will first discuss the zoom operators we included in Table 1b, and later report about different use of such operators by other authors.

Zoom-in_Detail shows on the screen a new view that represents with more details a portion of the current view. It has been described in Section 3 as essential operator of the VR, but it is available in all visual paradigms. Referring to Figure 2, the *Zoom-in_Detail* could be applied to one of the buildings representing a hospital ward to show a new view representing that specific ward. An early example of this operation has been proposed in the Spatial Data Management System (SDMS), where the user can navigate through the information space viewed on the large display screen, and access a data item by traveling to it in a virtual space and zooming into a subspace containing this data item and other data items, which is called porting in SDMS [2, 11].

Figure 3: Example of application of *Zoom-in_Subset*

Another type of zoom-in is called *Zoom-in_Subset*, since it allows to visualize all the subsets of a certain set. An example of such an operator is depicted in Figure 3: it is applied to *person* in order to show its subclasses.

The third type of zoom-in is called *Zoom-in_Fuzzy*, it is available in the extended form-based paradigm and allows visualizing exceptional cases through nested tables [9]. See an example in Figure 4.

Name	Birth date	...	Phone
John Smith	12/05/60	...	080/5572345
Adam King	04/11/56	...	080/5036032

Name	Birth date	...	Phone
John Smith	12/05/60	...	080/5572345
			080/5032756
			0330/348543
Adam King	04/11/56	...	080/5036032

Figure 4: Example of application of *Zoom-in_Fuzzy*

Two more operators are needed in visual information spaces, in order to manage the views on the screen. They do not involve database retrieval, but operate only on views, so that we have used the symbol V in the table cell. The first one is the *Magnify* operator, that sometimes is called zoom operator. It is used when, in order to accommodate a large view on the screen, the objects are shown very small. Therefore, to better see them, we magnify them, so that many more details will be visible. This operation is also available in the other paradigms. *Scroll* is the second operator acting only on the view. It is available in all paradigms.

When working with multimedia databases, similarity operators are worth considering. For example, in an image database, we can retrieve all images similar to a given one, if appropriate similarity functions are provided. In traditional database, a similarity condition is specified by conditions on the object attributes, therefore a similarity operator is actually the select operator of the relational algebra. Since we are dealing in this paper with traditional databases, we did not include such operators.

As we said in Section 3, the zoom operator is usually of two types: visual and semantic. We have called the former *Magnify*, while the three zoom operators we indicated in Table 1 are of semantic type. Another example of use of semantic zoom is in [18]. The concept of "elevation map" is interesting in that paper. The elevation indicates the user's perspective, i.e. it does not represent a physical elevation, rather it is a logical representation of the user's viewing distance from the visualized space. The elevation map specifies the operations that can be performed at a certain distance (elevation) from the objects. Therefore, the elevation map is used to control the availability of different operations as the user zooms in and out through the data space. At a certain elevation, the user may visualize some properties, while at other elevations the user may visualize other properties. For instance, if we look at a geographic map of the Europe, we can visualize the nations borders only, while, after zooming into some specific region of a nation, we can visualize all routes. This is similar to what we do when navigating in the VR paradigm. In the hospital example, the query operator *DM_Select* is applicable only when we are at a hierarchy level so that there is an object $s_i \in G(O)$; in other words, we can select an object that is the representation of an object in the database.

Visual zooming is discussed in [13], and a classification is proposed based on the effects the zooming has over the entire image. In a first case, when zooming use a lens moved on the image, the portion of the image magnified presents a sharp discontinuity at the boundaries, so users need to mentally integrate the two views. A second case refers to the use of fisheye lens, that avoids the previous inconvenience, since the selected object is magnified the most, and surrounding objects are progressively diminished in size, giving a perspective view. In a third case, a change of scale on one or all of the display axes is performed, causing information to leave or enter the viewing area. This has the advantage of uncluttering the screen, but big discrete jumps may cause disorientation in the users.

In VR literature, other concepts, such as portal, tunneling, etc., have been introduced to indicate the operation of changing universe during the interaction. Portal, as used for example in the WorldToolKit, is similar to our *Zoom_Detail* (it can be either *in* or *out*), and it allows partitioning the environment into multiple universes connected by portals. Tunneling generally involves changing from a multidimensional space to another [18], i.e. referring to a different database in a multidatabase system, while we are considering only interaction with one database.

5 Expressive power of the visual paradigms

On the basis of what we said in the previous section, we now discuss the expressive power of the various visual paradigms, by focusing primarily on VR. Looking at Table 1, form appears as the most powerful paradigm. The pioneer system QBE was shown to have expressive power even superior to the relational algebra [19]. Even if some diagram-based systems proposed in the literature were also more expressive than relational algebra [1, 8], diagram-based and icon-based systems are usually less powerful than form-based. Of course, they do not include the *Details* operator; indeed, the details of an object can only be visualized in a kind of form. Finally, the form-based paradigm and VR are not comparable.

It is worth noting that the purpose of Table 1 is to analyze the expressive power achievable in the various physical spaces. We studied the operations that can be executed, no claim is made on how easily they can be performed or how effective they will be, since this affects the system usability, that needs to be evaluated by appropriate means.

As it has been pointed out in [5], the goal of the people working with a visual query system is to retrieve the desired data. Such goal is usually accomplished through two main activities: 1) understanding the reality of interest, 2) formulating the query. In the former, the user should identify, possibly with the help of the system, the concepts that must be included in the query formulation. The first phase is significant and it is accomplished through a

navigation within the system. While the VR paradigm is very poor in retrieving data satisfying certain conditions, it is powerful in navigating within the system. In fact, VR techniques are introduced as a new interaction paradigm particularly suited to novice and/or occasional users, who generally formulate very simple queries. The users are facilitated in the database navigation since the system proposes them an environment that reproduces a real situation and gives the possibility of interacting by manipulating objects that have a direct correspondence with known objects. Moreover, VR scenarios are enriched with many details, taken from real contexts, so that users can easily locate the objects of interest; they can understand where they are, so reducing the risk of getting lost in an unknown system. We propose VR environments without pretending that users should be physically immersed in them, but with the aim of "visualizing reality" [15, 16], i.e. as a way to propose a visual environment closer to the reality of human beings much more than an icon-based visualization can achieve.

User interaction in VR is very similar to user behavior in the reality. For example, in a real hospital, if we look for data of patients, we go to a file cabinet where folders are kept, and look for a folder of the specific patient. When we find it, we open it and we look at the information of interest about that patient. We can look at the patient address or at the tests the patient did, etc. Let us think how this can be implemented in VR. We can show a scene including the file cabinet, select it (with *DM_Select*) to open it, then select a folder (again with *DM_Select*), so it appears as in Figure 5. As you can see, a set of attributes related to patient are indicated (others could be also shown). This is the kind of data retrieval that can be performed in VR. In order to allow the user selecting some attributes and/or specify conditions on them (i.e. to perform operations of the relational algebra), we should implement a system that properly combine VR and form-based paradigms.

Figure 5: An open folder with patient information

6 VR in progressive multiparadigm querying

In a previous paper [3], we have proposed a framework for multiparadigm visual access to databases, exploiting form-based, diagram-based and icon-based paradigms. The multiparadigm interface gives the user the possibility of switching among different visual interaction paradigms. In this paper we have formalized VR as a further interaction paradigm to be included in the multiparadigm framework. Another work proposing a multiparadigm approach for querying databases is described in [10].

Another interesting feature of a query system is allowing *progressive querying* [7]. Generally, non-expert or occasional users, who need to extract information from a database, are generally not able to directly formulate a query whose result fully satisfies their needs, at least in a first attempt. This is primarily due to the following reasons: a) the user does not know exactly what data are contained in the database and how such data are structured; b) he or she is not familiar with the query language and prefers to ask simple queries. Therefore, the user would prefer to formulate a complex query progressively, i.e. step by step, by first asking general questions, obtaining preliminary results, and then revisiting such outcomes to further direct the query in order to extract the result he or she is interested in. Note that a non-monotone query progression is allowed. During this process of *progressive querying*, an appropriate visualization of the preliminary result could provide a significant feedback to the user. First, the user has obtained an initial outcome and, even if it is not yet the expected final result, its visualization

could be a useful suggestion about the right way to proceed towards the ultimate query; otherwise, the user will immediately backtrack and try a different alternative. Conversely, if the user is satisfied with the result, he or she is also challenged to further investigate the database, so acquiring more information from it.

In a progressive querying system, a sequence of queries $Q_1, Q_2, \dots, Q_n$ represents a progressive query. Each Q_i refers to the step in which the query Q_i is being formulated. In a multiparadigm system, Q_i can be formulated in any of the interaction paradigms supported by the system, such as VR, form-based, diagrammatic, iconic, etc. The user can switch paradigms going from query Q_i to Q_{i+1} .

A progressive query is defined as $Q = Q_1 Q_2 \dots Q_n$, where each Q_i is a sub-query, and the meaning d_n of the progressive query Q is computed recursively as follows:

$$d_n = f(d_{n-1}, r(Q_{n-1}, Q_n)), n > 1$$

In other words, given the meaning d_{n-1} of the sub-queries $Q_1 \dots Q_{n-1}$, and the relationship r between Q_{n-1} and Q_n , we can compute the new meaning d_n of the progressive query $Q_1 \dots Q_n$ using the function f . The formalization allows some sub-queries to be VR queries, while the other sub-queries to be in other querying paradigms. For this to be possible, we must provide formal conditions for switching between paradigms.

The multiparadigm querying environment proposed in [3] translates the queries expressed in any visual paradigm into queries on a semantic data model called Graph Model [6], that is used as a common underlying formalism to which databases, expressed in the most popular data models, can be mapped. The algorithms to switch among visual paradigms are described on the base of the Graph Model formalism [4]. Therefore, the transformation algorithm from VR to other paradigms, and vice versa, will be also based on the underlying Graph Model representation, as described in the next section. The reader should refer to [3] for the details of the formalism.

As we discussed in the previous section, only a limited number of operations can be performed in VR. A query in VR is expressed by selecting, in the scene shown on the screen, one or more objects visible in that scene, i.e. objects belonging to $G(O)$. The selected objects are the *focus objects*. For example, in the hospital database, if we navigate to reach the patients room, we can only ask for some patient information, while we cannot ask for information related to both patient and lab-test, since lab-tests are visualized in a different scene. The strength of the VR query is that it enables the user to easily focus attention in the intuitive VR world and concentrate on a single focal object. But it may be difficult to express abstract relationships among objects in a VR query. Thus the primary strength of a VR query is also its primary limitation. This restrictive nature of VR queries motivates the combination of VR queries with other querying paradigms to form a progressive querying system.

A natural combination of progressive querying is VR and form-based queries. The user first formulates a VR query to locate the object, then switches to form-based query to specify certain conditions or relational constraints necessary in the reasoning process, and finally switches back to VR to continue the search. For example, in a virtual hospital the user first uses VR query to locate a patient, then switches to form-based query to find the lab tests that have been taken by the patient, and finally switches to VR query to locate the laboratories where the lab tests were conducted. Such progressive queries have high expressive power, and we are currently working on the theoretical problem to show that progressive queries combining VR and form-based queries are relationally complete.

7 Switching among visual paradigms

In order to include VR in the progressive multiparadigm querying system, we must provide the conditions for allowing a correct switching between paradigms. In this section, we discuss the transformation algorithms from VR to other paradigms, and vice versa; they are based on the underlying Graph Model representation. Referring to the Graph Model, in VR it is possible to manipulate only non-printable classes (entities), and no reference can be made to both printable classes and role nodes (domains and relations respectively). As a consequence, we cannot express conditions on printable classes and role nodes. More particularly, in VR we can express only queries involving one or more non-printable classes visible in the same scene.

A transformation from any paradigm to VR is possible only if the query is referring to database objects that are visualized in a same VR scene. On the other hand, going from VR to any paradigm is possible if in the correspondent Graph Model representation (also called GMDB) there is only one path connecting the nodes corresponding to the focus objects and the relations between them. The situation is different if we combine VR and form paradigms. But even in this case, to go from VR to another paradigm is straightforward, while the opposite is possible only if the query involves non-printable classes that are visualized in the same scene. For example, a query asking for information on patients who made a test can be translated to VR, while a query asking for information on

both patients and tests cannot be translated, since there is a non-determinism in selecting the corresponding scene in VR (patient or lab-test? see Figure 1).

As done for any paradigm in [3], a VR query is seen as a Graph Model query, since the Graph Model provides the underlying database representation in our multiparadigm framework. In [4] it has been shown how to switch among form-based, diagram-based and icon-based paradigms, exploiting the Graph Model representation. Therefore, to state the conditions for switching among all paradigms, including VR, we concentrate on the transformations from Graph Model query, expressed through the graphical primitives of such a model, to VR query (mapping GP_TO_VR), and vice versa (mapping VR_TO_GP).

In [3], the atomic queries were queries that could be translated in a different paradigm. The atomic queries were defined formally as queries whose corresponding GMDB were admissible, and the admissibility conditions were given. In order to characterize a VR atomic query, in the following Lemma we give the similar admissibility conditions in terms of GMDB. Note that we use the symbol $\supset$ to indicate the relation *includes* shown in Figure 1. As in [3], given a GMDB, N_{Cu} is the set of non-printable class nodes, N_R is the set role nodes, and AD is the set of adjacent nodes, f_3 is a total function mapping each node to one value in $\{unselected, selected, displayed\}$. Moreover, T is the mapping from N_{Cu} to the set O of objects in the database.

LEMMA

A GMDB D is admissible if either one of the following conditions holds:

1. $\exists \mid n_i \in N_{Cu}$ in D such that $f_3(n_i) = selected$;
2. if $n_{i1}, n_{i2}, \dots, n_{ik} \in N_{Cu}$ in D are such that $f_3(n_{ij}) = selected, j = 1, \dots, k$, then the following conditions must hold:
 - a. $\exists s \in S$ such that $s \supset s_{i1} = G(T(n_{i1}))$ and $s \supset s_{i2} = G(T(n_{i2}))$ and ... and $s \supset s_{ik} = G(T(n_{ik}))$;
 - b. $\forall n_{ip} \neq n_{ir}, p, r = 1, \dots, k$, if $\exists q_t \in N_R$ in D such that $\{q_t\} = AD(n_{ip}) \cap AD(n_{ir})$ then $f_3(q_t) = selected$;

In other words, the above Lemma formalizes that an atomic query in VR is one involving one non-printable class (condition 1) or more non-printable classes that in VR are visible in the same scene (condition 2a) and also there is only one path connecting such non-printable classes and the relations between them (condition 2b).

In order to describe the transformation from Graph Model query to VR query, and vice versa, let us characterize the interaction mechanisms of the VR paradigm through the graphical primitives, first introduced in [6]; they allow to express any query-oriented user interaction with a database in terms of two simple graphical operations: the *selection of a node* and the *drawing of a labeled edge*. In VR we cannot set conditions, the only possible operation is the selection of one or more objects of a same scene. Therefore, the mappings GP_TO_VR and VR_TO_GP will only involve the selection of a node.

The mapping VR_TO_GP establishing the semantics of the VR interaction mechanism is defined as follows:

1. the selection of an object $s_i \in G(O)$ in VR corresponds to the selection of the node $n_i = T^{-1}(G^{-1}(s_i)) \in N_{Cu}$;
2. if two objects $s_i, s_j \in G(O)$ have been selected, then $f_3(q) = selected$, where $q \in N_R, \{q\} = AD(n_i) \cap AD(n_j), n_i = T^{-1}(G^{-1}(s_i)) \in N_{Cu}, n_j = T^{-1}(G^{-1}(s_j)) \in N_{Cu}$.

The semantic for the mapping GP_TO_VR is the following:

1. the selection of a node $n_i \in N_{Cu}$ corresponds to the choice, in VR, of the scene in which $s_i = G(T(n_i))$ is visualized;
2. the selection of nodes $n_i, n_j \in N_{Cu}$ and $q \in N_R$, where $\{q\} = AD(n_i) \cap AD(n_j)$, corresponds to the choice, in VR, of the scene in which both $s_i = G(T(n_i))$ and $s_j = G(T(n_j))$ are visualized.

From the above, we can easily derive the algorithms to transform a VR query to a Graph Model query, and vice versa.

8 Conclusion

With the explosive growth of the information technology, the interests are high in providing powerful and yet simple-to-use tools for querying the databases. VR query, while too “concrete” for sophisticated applications, may be a very powerful paradigm if used in combination with other query paradigms.

We have shown that a natural combination of progressive querying is VR and form-based queries. The user first formulates a VR query to locate the object, then switches to form-based paradigm to specify certain conditions or relational constraints necessary in the reasoning process, and finally switches back to VR to continue the search.

Ideally, the progressive querying system may be able to choose the representation paradigm for both query and result on the basis of the knowledge it has about the user, the task and the application domain. In practice, the user should always have the freedom to change the paradigm. With this conceptual framework, we are currently investigating the heuristics and strategies for progressive querying using a combination of VR queries and form-based queries.

9 References

- [1] Angelaccio M, Catarci T, Santucci G. QBD*: a graphical query language with recursion. *IEEE Transaction on Software Engineering* 1990; 16: 1150-1163
- [2] Bolt R. Put-that-there: voice and gesture at the graphics interface. *ACM Computer Graphics* 1980, 14: 262-270
- [3] Catarci T, Chang SK, Costabile MF, Levialdi S, Santucci G. A graph-based framework for multiparadigmatic visual access to databases. *IEEE Transaction on Knowledge and Data Engineering*, to appear
- [4] Catarci T, Chang SK, Santucci G. Query representation and management in a multiparadigmatic visual query environment. *Journal of Intelligent Information System* 1994, 3: 299-330
- [5] Catarci T, Costabile MF, Levialdi S, Batini C. Visual query systems for databases: a survey. Technical Report SI/RR 95/17, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", Italy, 1995
- [6] Catarci T, Santucci G, Angelaccio M. Fundamental graphical primitives for visual query language. *Information System* 1993, 18: 75-98
- [7] Chang SK, Costabile MF, Levialdi S. Reality bites - progressive querying and result visualization in logical and VR spaces. In: *Proc. of IEEE Symposium on Visual Languages*. St. Louis, 1994, pp 100-109
- [8] Cruz IF, Mendelson AO, Wood PT. G+: recursive queries without recursion. In: *Proc. of the 2nd International Conference on Expert Database Systems*. 1988, pp 355-368
- [9] Dreizen H, Chang SK. Imprecise database schema: a rationale for relations with embedded subrelations. *ACM Transactions on Database Systems* 1989, 14: 447-479
- [10] Doan DK, Paton NW, Kilgour AC, al-Qaimari G. Multi-paradigm query interface to an object-oriented database. *Interacting with Computers* 1995, 7: 25-47
- [11] Herot CF. Spatial management of data. *ACM Transactions on Database Systems* 1980, 5: 493-514
- [12] Ignatius E, Senay H, Favre J. An intelligent system for task-specific visualization assistance. *Journal of Visual Languages and Computing* 1994, 5: 321-338
- [13] Jog N, Shneiderman B. Starfield information visualization with interactive smooth zooming. In: *Proc. of the Visual Database Conference '3. Lausanne*, 1995, pp 1-10
- [14] Levialdi S, Massari A, Saladini L. Visual metaphors for database exploration. In: *Proc. of the Second FADIVA Meeting*. Glasgow, UK, 1995
- [15] Preece J, Rogers Y, Sharp H, Benyon D, Holland S, Carey T. *Human-computer interaction*. Addison-Wesley, Wokingham, England, 1994, pp 207-209
- [16] Robertson GG, Card SK, Mackinlay JD. Nonimmersive virtual reality. *IEEE Computer* 1993, 26: 81-83
- [17] Tsuda K, Yoshitaka A, Hirakawa M, Tanaka M, Ichikawa T. IconicBrowser: an iconic retrieval system for object-oriented database. *Journal of Visual Languages and Computing* 1990, 1: 59-76
- [18] Woodruff A, Su A, Stonebraker M. Navigation and coordination primitives for multidimensional visual browsers. In: *Proc. of the Visual Database Conference '3. Lausanne*, 1995, pp 323-331

- [19] Zloof MM. Query-by-Example: operation on the transitive closure. Technical Report RC5526, IBM Research, Yorktown Heights, 1976