24

An automatic visual database interface

E. Pichat
Université Claude Bernard - Lyon 1
Bâtiment 710 43 boulevard du 11 novembre 1918
France 69622 Villeurbanne Cedex
Tel. 72 44 83 70 Fax 72 44 83 64 e-mail: pichat@ligia.univ-lyon1.fr

D. Saker
Nat Systèmes
100 rue La Fayette France 75010 Paris
Tel. (1) 40 22 95 94 Fax (1) 40 22 95 92

Abstract

Today's relational and object-oriented databases come with very powerful declarative query languages like SQL. However, such languages are still not adapted to the needs of the occasional user because of their syntax requirements and because they do not facilitate the understanding of the database semantics, navigation within tables and the formulation of queries.

This paper proposes an especially assisting visual interface for database query. This is possible with a powerful and easy-to-understand database scheme representation, the *normalized semantic graph (NSG)* which is underlain by the *Universal Relation with Inclusions (URI)* data model, and by making NSG more explicit by automatic generation. First the interface displays the set of the relational schemes and a set of basic links in background: it is the *relation-to-relation NSG*, a kind of restriction of NSG. Displaying links with tables makes it easier for the user to understand the database scheme and to find his way about tables. Secondly the interface calculates and displays the possible join-links of a table designated by the user. This is the *evolving query graph*. As it evolves with the designated node, the user progressively specifies his query: he only has to select the attributes to project, precise selection conditions and click on the desired join-links to obtain the *query graph*, which is translated in SQL.

This interface by-passes syntactic specifications and assists the user as closely as possible in his declaration. The paper also shows how to generate all the possible links between relations implied by the NSG as well as how to complete them with compatible attributes. In this way, the *complete query graph* is obtained. But to avoid drowning the user in a mass of information, only those links of the complete query graph which are contiguous to the node designated by the user are displayed.

A prototype is available in the GraphTalk® environment.

Keywords

Data model, universal relation with inclusion, normalized semantic graph, common attribute, compatible attribute, inclusion dependency, complete query graph, evolving query graph

S. Spaccapietra et al. (eds.), *Visual Database Systems 3*

1 INTRODUCTION

DBMS are today popular applications. But unfortunately, users are still reduced to asking predefined questions or to making the effort of using data manipulation languages such as SQL. Although SQL is a declarative and much easier language to use than, say, network and hierarchical model query languages, it still remains a difficult language to learn for the non-expert user [Reisner 81] and does not sufficiently enhance productivity. To formulate a query, the user is expected to know the theory underlying the data model, remember the database scheme learn, specify join conditions and force the syntax of the query language.

Considering the limited number of tables, attributes and syntactic units likely to be involved in querying a database and the advantages accruing from the graphical representation of the conceptual data scheme, visual query interfaces represent an interesting alternative to formal query expressions. This is even more so with the emergence of new techniques of Man-Machine interaction (direct manipulation, WYSIWYG and object-action dialog) that are more effective from the dual point of view of learning and use.

The table in Figure 1 provides a quick summary of the evolution of graphical interfaces aimed at assisting users in database query. The assistance provided by CUPID is very limited because the user is still expected to link the different components and maintain coherence with the data scheme himself. CUPID has served as model for many interfaces, such as LAGRIF [Lakhal 86] and IQL [Ramos 91]. ISIS represents an important step forward in the effort to assist users in query formulation: before ISIS, join paths were not made explicit to the user and syntactically correct but semantically incorrect queries were possible. Its graphical conceptual scheme, based on the semantic model SDM, generates a graphical representation of classes of objects grouped together under several trees using two types of relationship: superclass/subclass and aggregation. GLAD also represents a major step forward since it visualizes a unique graphical conceptual scheme of the database which it uses as query medium. In addition, its implementation with an object-oriented language testifies to the blessings accruing from the object paradigm in the implementation of Man-Machine dialog. The first interfaces to implement direct manipulation [Schneiderman 83] [Nanard 90] appeared in the 90s. The all-graphical approach to the specification of complex selection conditions in Pasta-3 and other interfaces such as IQL, is only justified as a component of a graphical toolbox and cannot be a substitute to the formal representation of statements. Visual interfaces also facilitate the querying of object-oriented DBMS with their inheritance and composition links. CANDID, CQL and SUPER permit the modeling of complex data. OHQL authorizes declarative queries formulated with methods and navigation operations drawn from the hypertext approach.

The user interface proposed in this paper implements the enhancements described above:

- it is descriptive in so far it provides a concise and complete visualization of the data scheme (only one medium is necessary);
- it uses the same medium both for the description of the data scheme and for the representation of the formulated query: the semantic assistance given to the user cannot be more complete;
- it is interactive: the formulation of a query is made by simply designating the nodes and arcs of the displayed semantic graph and the syntactic units through the technique of direct manipulation;
- it accepts both complex [Fallouh 94] and normalized data although because of problems of space, we will restrain ourselves here to only normalized data.

Our interface introduces three enhancements:

First, it uses a more powerful and canonical data model than the usual semantic models. In Section 2, we introduce the *Universal Relation with Inclusions* model adopted in URITalk interface, especially its graphical representation, the *normalized semantic graph.*

The only available running joins at the present state of development of DBMS and data manipulation languages concern two tables as elementary join conditions. The second advantage of the query interface presented here is that it does not only display relation-to-relation links of the normalized semantic graph but also helps the user in the formulation of

queries by being able to visualize all the possible binary semantic links. Section 3 is devoted to the construction of the *complete query graph* from the normalized semantic graph. However, the complete query graph cannot be displayed because of the large number of links.

Query interface \ Visualization of	tables	join symbol	conceptual schemes	query graph	one conceptual scheme	conceptual scheme support query
SQL [IBM 81] [Melton 93]	no					
QBE [Zloof 77]	yes	no				
CUPID [McDonald 75]		yes		yes		
RU [Ullman 89]			no			
ISIS [Goldman 85]			yes	no		
Pasta-3 [Kuntz 90]				yes	no	no
GLAD [WU 84] [WU 89]					yes	yes
CANDID [Trepied 89]						
CQL [Kari 90]						
SUPER [Auddino 91]						
OHQL [Andonoff 92]						
URITalk [Saker 93]						

Figure 1 Evolution of query interfaces.

Section 4 presents a partial graph of the complete query graph which is used as query medium: the *evolving query graph* that continues to change with every progress the user makes in the specification of his query.

Finally, Section 5 shows how the URITalk interface assists the user in formulating queries involving selection-projection-joins on the evolving query graph. Thus the *query graph* is obtained.

2 THE URI DATA MODEL AND THE NSGraph

2.1 The Universal Relation with Inclusions

Relational DBMS and relational query languages are based on the relational data model [Codd 70] [Ullman 89]. The *Universal Relation* (UR) assumption provides a rigorous procedure for constructing the different relations of a database. The *functional dependencies* (FD) $\underline{L}$ R, where L (the *left side*) and R (the *right side* or functionally determined attributes side) are sets of attributes that belong to the UR **R**(U), permit the decomposition of the U into a set of relational structures $\mathbf{R}_i(\underline{K_1}\ \underline{K_2}\ ...\ \underline{K_n}$ Remainder) in refined third normal form [Maier 83] through the normalization process. K_1, K_2 ..., K_n are a set of keys of $\mathbf{R}_i$ and Remainder the subset of the set of the attributes of $\mathbf{R}_i$ that do not belong to any of the keys. Let be X^+ the *closure* of the attribute set X (X^+ is the set of the attributes functionally determined by X). Then each relation closure U_i^+ and the number of keys per relation are invariants of the normalized relational database scheme. The introduction of *join components* (JC), assimilable to FDs without right side, for the purpose of preserving data, increases the descriptive power of the model.

However data modeling remains unsatisfactory with respect to Entity-Relationship, for example. Inclusion dependencies make up for this shortcoming: an *inclusion dependency* (ID) [Casanova 84] [Lafaye 1982] [Mannila 86] [Mitchell 83] noted:

$A_1 \ldots A_k \subseteq_\alpha B_1 \ldots B_k$

where A_i and B_i are attributes and α names the ID and distinguishes it from the set inclusion, is verified on the Universal Relation **R** if and only if the projections of any instance R of **R** on the right and left sides of the ID verify the set inclusion:

$R[A_1 \ldots A_k] \subseteq R[B_1 \ldots B_k]$.

The ***Universal Relation with Inclusions*** (***URI***) is the Universal Relation data model extended to the IDs [Pichat 90]. URI comes with a normalization process extended to IDs [Pichat 89]. The above-mentioned results obtained for the normalized form of UR are valid for URI.

The set of normalized relational structures is partitioned with the concept of ***Partial Universal Relation*** (PUR). Two normalized structures, $\mathbf{R}_i$ and $\mathbf{R}_j$, belong to one and the same PUR if and only if there is a sequence of normalized relational structures $\mathbf{R}_i,\ldots\mathbf{R}_k,\mathbf{R}_1,\ldots,\mathbf{R}_j$ such that for any pair $\{\mathbf{R}_k, \mathbf{R}_1\}$ of adjacent relational structures in the sequence, their closures U_k^+ and U_1^+ have at least one attribute in common. The PURs also partition the set U of the attributes of **R**.

2.2 The Normalized Semantic Graph

The ***normalized semantic graph*** (***NSG***) is the URI graphical representation of a database scheme. It is made up of a set of nodes and a set of hyperarcs (see Appendix 1):

- the nodes are PURs. Each PUR can, in turn, be represented with a directed graph without circuits:
  - its nodes are the normalized relational structures $\mathbf{R}_i(\underline{K}_1\ \underline{K}_2 \ldots \underline{K}_n$ Remainder);
  - the arc $(\mathbf{R}_i, \mathbf{R}_j)$ exists if and only if
    - the closure U_i^+ at the origin includes the closure U_j^+ at the extremity,
    - it cannot be obtained through the transitivity of the other arcs,
    - its origin and its extremity have at least one attribute in common: $U_i \cap U_j \neq \phi$.

    These closure inclusion arcs are termed ***common attribute arcs***; they represent monovalued functions,
- the hyperarcs represent the normalized inclusion dependencies. They are termed ***inclusion hyperarcs between PURs*** because they are graphically represented from a set of (initial) PURs to another set of (terminal) PURs (see 3.2-a below).

Example: The figure in Appendix 1 shows the NSG of the "the Program Committee's treatment of submitted papers" [Heuser 93], itself part of the classical example "Preparation of an IFIP conference" [Olle 82]. Each paper, identified by its title, may concern a number of different conference topics and be co-authored by several persons. Each referee has one address, is competent in several topics and cannot receive more than one paper at a time. As soon as a paper is read, the referee writes his report and the paper is considered to have changed from the table of distributed papers to the table of papers that have been evaluated. All the papers are judged together. The NSG in Appendix 1 has five nodes (PUR) and six IDs. Three of the nodes are made of only one relation each (**JudgedPaper**, **Evaluation** and **Distribution**). The two other PURs (RECEIVED PAPERS and REFEREES) contain two

relations each (**Author** and **PaperTopic**, on the one hand, and **RefereeTopic** and **Referee**, on the other). REFEREES contains the common attribute arc (**RefereeTopic**, **Referee**) showing that the closure of the **RefereeTopic** relation includes the closure of the **Referee** relation, that is a tuple of the **RefereeTopic** relation identifies one and only one tuple of the **Referee** relation. We also notice that the **RefereeTopic**, **PaperTopic** and **Author** relations are join components. With the exception of the relation **Referee** which has two keys (RefereeName and RefereeNickName), all relations have a unique key.

The underlying principle of semantic models is to provide concepts that are powerful enough to yield the closest possible specification of a real-world situation. The URI model is more powerful than most Entity-Relationship models because of its underlying UR assumption and IDs [Andraws 91]. It provides the framework for bringing together OO data model composition and inheritance, relational database key and referential dependency, and most Entity-Relationship model extensions. It is used in this article to calculate (and visualize) the semantic relationships between tables based on common attributes and IDs.

3 FROM NSG TO COMPLETE QUERY GRAPH

In this section we examine more closely the idea floated and experimented upon by [Wu 86] concerning the formulation of queries: instead of using a pure relational database scheme (or set of relations without links), we construct a query medium defined by the data semantics.

Of course URITalk vizualizes the explicit binary links of NSG (its common attribute arcs) and the inclusion arcs between relations easily deduced from its inclusion arcs $L \subseteq_{\alpha} R$ between PURs (case of two PURs having either only one relation intersecting L or R, or a relation functionally equivalent to L or R). The graph thus obtained is called the ***relation-to-relation NSG***: it is a kind of partial graph of the NSG, but a graph between relations and consisting only of arcs of the NSG, making it easier to identify relations to be joined.

An area in which URITalk innovates is that it completes the relation-to-relation NSG with edges:

- which are common attributes not represented by common attribute arcs, or
- which are implicit because they are represented in NSG by n-ary IDs with n>2 or binary IDs concerning more than two relations, or
- which result from compatibility, or
- which are transitive links (for instance, Appendix 3, the two inclusion arcs JudgedPaper $\subseteq_7$ Paper).

We examine in this section how to construct the ***complete query graph*** (with all the possible table-to-table relationships) from the basic relation-to-relation NSG. The expression of join conditions for users becomes simply a matter of selecting the appropriate links. In the next section, we will see that the complete query graph is not visualized in its wholeness, but through the evolving query graph.

3.1 Common attribute edges

We have seen in 2.2 that the NSG visualizes common attribute arcs. However it does not visualize all the possible joins implied by common attributes. For instance the **Author** and **PaperTopic** relations of Figure 2-a have the common attribute Paper. Yet, the many-many relationship between them resulting from the common attribute is not visually represented on the semantic graph. The common attribute edge between **Author** and **PaperTopic** (see Figure 2-b) is the medium of the natural join between the two relations and is part of the complete query graph.

(a) *Normalized semantic graph (extracted from the NSG in Appendix 1)*

(b) *Normalized semantic graph with common attribute edge*

Figure 2 Common attribute edge.

3.2 Representation of an inclusion hyperarc with relation-to-relation arcs

We will explain this representation in three steps:

a) From an inclusion arc on the UR to an inclusion hyperarc between PURs

Generally speaking, IDs are defined on the UR (see Figure 3-a). However, given that an attribute cannot belong to more than one PUR, it is easier to understand the representation of an ID between semantic graph PURs. For example, the inclusion dependency:

TestPilot Prototype $\subseteq_1$ Driver Car

is easier to visualize in Figure 3-b than in Figure 3-a. An ID L $\subseteq_\alpha$ R will be represented by a hyperarc whose origins are PURs having an intersection that is not empty with L, and whose extremities are PURs having an intersection not empty with R. Let us emphasize that a PUR can be both origin and extremity of a hyperarc.

b) From an inclusion hyperarc between PURs to an inclusion hyperarc between relations

Because relational DBMS do not implement the PUR concept, we need to represent IDs between relations to be able to use the links resulting from the IDs as possible join paths between relations.

Figure 3-c illustrates this representation in the form of an inclusion hyperarc between relations. Each ID between PUR: L $\subseteq_\alpha$ R (from the set of the source PURs of α $\mathbf{P}_{URl}$ to the set of the terminal PURs of α $\mathbf{P}_{URr}$) can be represented by an ***inclusion hyperarc between relations***, from the relations $\mathbf{R}_{lk}(U_{lk})$ belonging to a PUR $\mathbf{P}_{URl}$ and having the not empty intersection ($U_{lk} \cap L$) with L, to the relations $\mathbf{R}_{rm}(U_{rm})$ belonging to a PUR $\mathbf{P}_{URr}$ and having the not empty intersection ($U_{rm} \cap R$) with R. In this way we obtain a NSG represented with only relations $\mathbf{R}_i(\underline{K}_1\ \underline{K}_2 \ldots \underline{K}_n$ Remainder) defined Section 2 and each hyperarc is a common attribute arc or an inclusion hyperarc defined between relations. We shall be using the term ***NSG between relations*** to designate this representation of NSG in the rest of this paper.

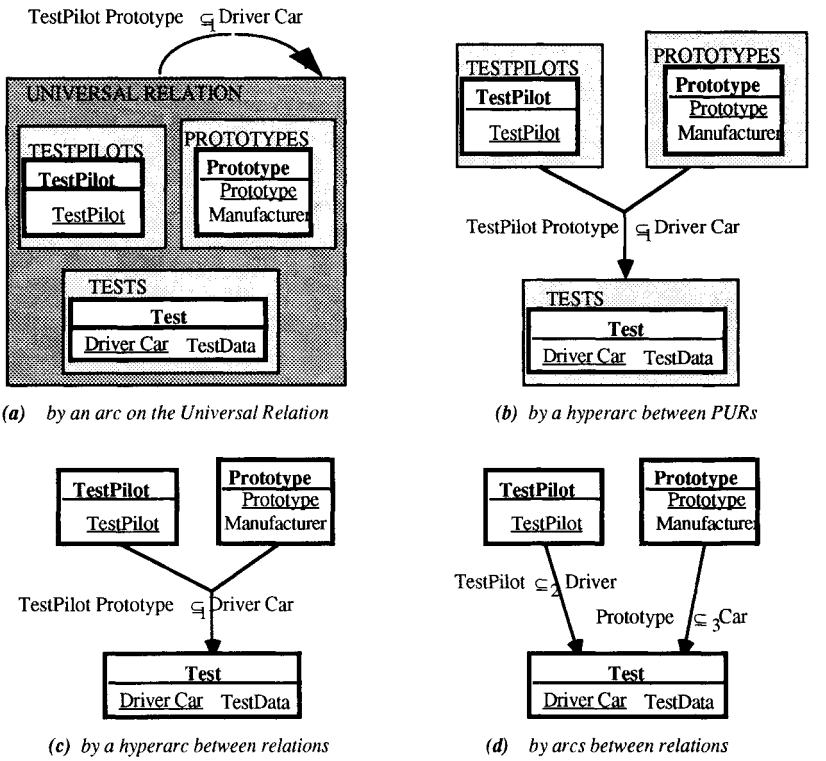


Figure 3 ID representations.

c) From an inclusion hyperarc between relations to a set of relation-to-relation inclusion arcs

If the representation of an ID by a hyperarc between relations is semantically equivalent to its definition between PURs, it does not bring out the relation-to-relation connections implied and so does not assist the user. Before specifying them, let us introduce the notations we shall be using: given an ID $L \subseteq_\alpha R$ and L' a sub-sequence of L, $\prod_\alpha(G')$ refers to the subset of R corresponding to the subset L'. In the same way, $\prod_{-\alpha}(R')$ is the sub-sequence of L corresponding through α to the sub-sequence R' of R. For example, given the ID ABCEF $\subseteq_1$ HIJKM, and the set X = BCFIMN, we then obtain:

$$X \cap L = BCF \text{ et } \prod_1(X \cap L) = IJM,$$
$$X \cap R = IM \text{ et } \prod_{-1}(X \cap R) = BF.$$

Any ID $L \subseteq_\alpha R$ from the PUR $\mathbf{P}_{URl}(U_l)$ to the PUR $\mathbf{P}_{URr}(U_j)$, or hyperarc from the relations $\mathbf{R}_{lk}(U_{lk})$ to the relations $\mathbf{R}_{rm}(U_{rm})$, is represented for querying by the set of relation-to-relation inclusion arcs obtained by:

For all relation $\mathbf{R}_{lk}(U_{lk})$ of $\mathbf{P}_{URl}(U_l)$ such that $(L \cap U_{lk}) \neq \phi$ do
  For all relation $\mathbf{R}_{rm}(U_{rm})$ of $\mathbf{P}_{URr}(U_r)$ such that $(R \cap U_{rm}) \neq \phi$ do
    If $R' = \prod_{\alpha}[L \cap U_{lk}] \cap U_{rm} \neq \phi$ then
      Create the inclusion arc $\prod_{-\alpha}(R') \subseteq_{\alpha lkrm} R'$ of $\mathbf{R}_{lk}(U_{lk})$ to $\mathbf{R}_{rm}(U_{rm})$.

For example, the ID of Figures 3-a, b and c is represented by the two relation-to-relation inclusion arcs shown in Figure 3-d.

The representation of an ID by a set of relation-to-relation inclusion arcs is enough for querying tables but it degrades the semantics of the original ID. This is why updating operations entails checking data consistency by calculating joins in addition to checking relation-to-relation inclusion arcs [Fallouh 91].

3.3 Edges of compatible attributes

The concept of ID $A \subseteq_{\alpha} B$ generalizes the concept of common attribute. We can go ahead and generalize the inclusion dependency: two attributes are said to be *compatible* if their domains (or the set of all possible values) are not disjoint and if they are "semantically close". For example, knowing that sometimes workers decide to go on further studies and that students are sometimes engaged in paid employments, we can say that StudentName and EmployeeName are compatible. On the contrary, the attributes PersonName and PlaceOfBirth are, *a priori*, not compatible even if it is possible for someone to be known as Paris. However, if we are working on the etymology of names, then PersonName and PlaceOfBirth will be compatible attributes.

A *compatibility relation* $Ai \Leftrightarrow Aj$ defined on all the attributes of the database scheme is an *equivalence relation*. Such a relation therefore defines a partition of the attributes of the database scheme into equivalence classes or *compatibility classes*.

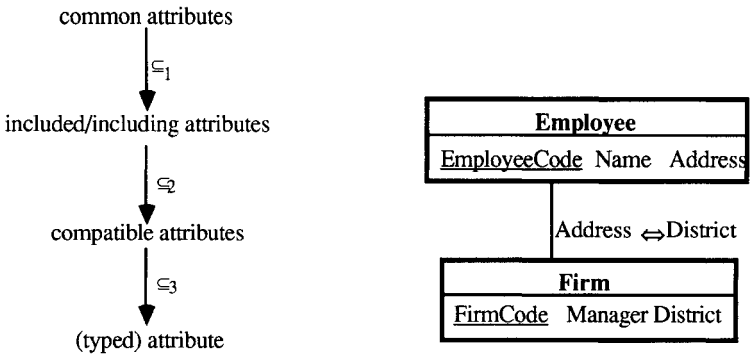
There may be several compatibility relations on the attributes. It is also possible to define a compatibility relation between the compatibility classes, that is, create more encompassing compatibility classes from the basic compatibility classes. Consider for example the following relations:

$\mathbf{R}_1$(Person Name Passion)
$\mathbf{R}_2$(Person Occupation)
$\mathbf{R}_3$(Student Job Hobby)
$\mathbf{R}_4$(Secretary Duty)
$\mathbf{R}_5$(CivilServant Function).

All the attributes of these relations can be partitioned into four compatibility classes:

C_1 = {Person, Student, CivilServant, Secretary}
C_2 = {Hobby, Passion}
C_3 = {Occupation, Job, Duty, Function}
C_4 = {Name}.

If we take the ID Student $\subseteq_1$ Person, CivilServant $\subseteq_2$ Person, Secretary $\subseteq_3$ Person, the class C_1 of compatible attributes coincides with the set of all the attributes of the same rank in an ID. Generally speaking, the common attribute relation defined on all occurrences of the attributes of a database scheme is more discriminating than the inclusion relation between attributes, which in turn is more discriminating than the attribute compatibility relation, which is also more discriminating than the relation ensuing from the fact that two attributes are of the same type (see Figure 4-a). For the class C_2, a passion, for example music, can be a hobby, etc. Compatibility classes C_2 and C_3 can be grouped together in a second-order class to give a compatibility class C_{23} comprising the join possibilities defined by C_2 and C_3. The visual interface may be consulted in the restricted compatibility relation or in the extended form.



(a) Inheritance relationship between pairs of attributes used in the article

(b) Edge of compatible attributes

Figure 4 Compatible attributes.

An edge of compatible attributes links two tables with compatible attributes: such an edge visualizes a possible join between the two tables. For example, Figure 4-b shows the edge of compatible attributes (Address ⇔ District) for retrieving the names of employees who come from the same town as a certain firm.

4 EVOLVING QUERY AND QUERY GRAPHS

The complete query graph developed in the preceding section will in complex cases be like a spider's web. That is why URITalk displays only a partial graph, the *evolving query graph.*

4.1 Links contiguous to a node in the evolving query graph

The nodes of the evolving query graph are those of the relation-to-relation NSG, and its arcs are those of the relation-to-relation NSG plus, for a node designated by the user, the contiguous links of the complete query graph. The mechanism of navigating through and displaying new links in the evolving query graph is simple. The user moves from one node to another by pointing to the destination node. For a given designated node, the ***evolving query graph*** visualizes, with the relation-to-relation NSG as background, the complete query graph links that have this node as one of their extremities. The user selects some of these links with the result that they become incorporated in the ***query graph.*** Thereafter, he designates another node of the evolving query graph: the contiguous links of the complete query graph are in turn displayed whereas the contiguous links of the first node not belonging to the relation-to-relation NSG or to the query graph disappear from the query window (see the illustration in 5.3).

4.2 Extension of the query graph to autojoins, nested joins and group operators

The query graph greatly facilitates the user's task of specifying the joins of different tables. However, the definition of the join of a table by using the same table (*autojoin*) requires the duplication of the table [Ramos 91] and the creation of links between the two [Wong 76]. The formulation of a sub-query or a query linked to another query by UNION, INTERSECT or

MINUS group operator is done in an auxiliary window for duplicating evolving query graph and query graph.

Figure 5 shows the different graphs considered in the article.

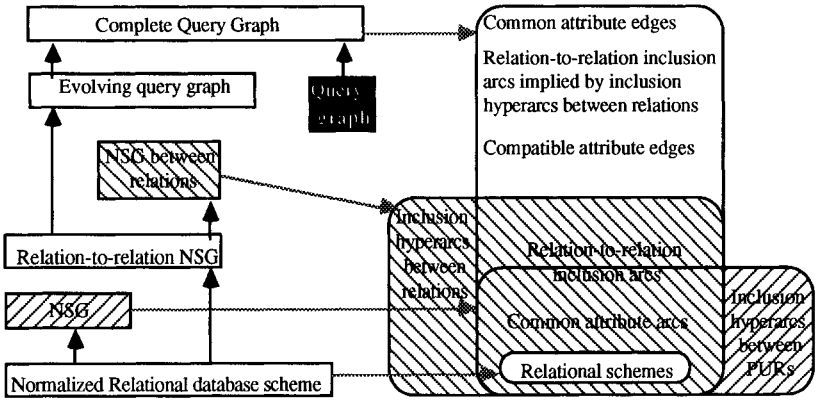


Figure 5 The different graphs considered in the article, ordered by the partial subgraph relation ↑, their nodes and links.

5 DESCRIPTION OF URITalk INTERFACE

The URITalk interface facilitates the formal and visual conception of the database scheme as well as the graphical querying of the database. It was developed in the object-oriented environment GraphTalk® [Jeulin 88] for software engineering workshop development.

5.1 The conception of the database scheme

The conception of the database scheme from functional dependencies, join components and inclusion dependencies is done with the help of the LACSI package [Roche 90]. The software package has two modules:

- a specification interface for entering and modifying FDs and JCs in a "Functional-Dependencies" window, and IDs in an "InclusionDependencies" window,
- a normalization module for fine-tuning the normalization (extended to ID) of the specification. It produces a NSG (for example, the figure in Appendix 1)

Each relation is an object represented by a rectangle named after the relation. The relation rectangle (see Figure 6) has a pull-down menu for performing the following tasks:

- *display*ing or *hide*-ing the rectangles representing keys and attributes of the relation that do not belong to a key,
- *contiguity* for searching for links that are contiguous to the relation (this operation can also be carried out by double-clicking above the relation icon),
- *rename*-ing the relation.

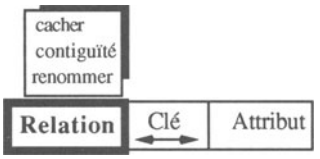


Figure 6 Relation icon.

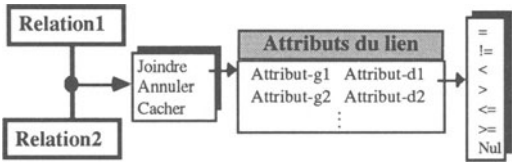


Figure 7 Link icon.

Each ID link also has a pull-down menu which, aside from permitting the management of the graphical representation, facilitates the definition of a join condition on all the pairs of attributes of the same rank associated with the link. The "Null" option of the comparator menu, selected for a line in the list, enables the exclusion of a pair of attributes of the join condition. The "Undo" item is for undoing the join operation associated with a link. Finally, the "Hide" item in the menu hides a link that belongs to the evolving query graph without belonging to the relation-to-relation NSG. This has the effect of augmenting the readability of the scheme.

5.2 The graphical query screen

The interface has one main window:

- the **query window**, which displays three superimposed graphs with different shades:
  . the *relation-to-relation NSG* is bright and fixed, in the background to guide the user,
  . the *evolving query graph* is less bright, contains in addition to the first graph (the same nodes and arcs) all the binary links between a node selected for query and the other nodes,
  . the *query graph* under development, which is in reverse video.
  The window has a selection bar which, in addition to the traditional *File* and *Edit* menus, contains:
  1. the *Query* menu, used for choosing the type of query to be made on the query graph (default query, insert, modify, delete). To make an insert operation, the user simply designates the appropriate relation and the corresponding edit window is displayed. In the same vein, to delete or update, the user simply designates the relation concerned and enters a selection condition. The menu has the following three sub-menus:
     . Condition: for specifying a condition in WHERE or HAVING clauses or UPDATE or DELETE commands in a new query window,
     . Set operator: for formulating queries involving the three operators,
     . Execute: generates the SQL code and, if it is coupled with a DBMS, executes the query thus generated.
  2. the *Help* menu.

The interface comes with three other windows:

- the **attribute window** for displaying all the attributes of the database scheme,
- the **SQL code window** for displaying the corresponding SQL code of the visual query formulated,
- the **result window** for displaying the result of the executed query.

5.3 Illustration

Let us use the NSG in Appendix 1 to show how to formulate the following query: "find the addresses of possible readers of the papers written by John". The user will proceed thus:

The initial screen contains the relation-to-relation NSG (Appendix 2).

The user designates the projected attributes: **RefereeName** in either **RefereeTopic** or **Referee** (see Appendix 4), RefereeAddress in the relation **Referee**. To establish the selection condition Author = 'John', the user carries out the following actions:

- double-clicks on the attribute Author of the **Author** relation and the pull-down menu associated with this attribute is displayed,
- chooses the *Selection* option in the menu,
- (implicit) chooses the comparator "=",
- (implicit) chooses the option "value to be entered",
- enters the value 'John' in the open edit box.

In order to formulate the join predicates, assuming the user has selected the **RefereeTopic** relation, the user double-clicks above its icon. The system will, in response, display the contiguous links with **RefereeTopic** (that have not yet been displayed) of the complete query graph, in this case the two inclusion arcs:

Evaluation.EvalRefereeName $\subseteq_2$ **RefereeTopic**.RefereeName,
Distribution.DisRefereeName $\subseteq_4$ **RefereeTopic**.RefereeName.

Double-clicking on the common attribute arc RefereeName from **RefereeTopic** to **Referee**, causes it to belong to the query graph, that is, specifies the natural join predicate:

Referee.RefereeName = **RefereeTopic**.RefereeName.

In the same vein, double-clicking on the arc of the ID PaperTopic $\subseteq_6$ RefereeTopic specifies the equijoin condition:

PaperTopic.PaperTopic = **RefereeTopic**.RefereeTopic.

Next the user selects the **PaperTopic** relation (see Appendix 5) and the **RefereeTopic** relation is unselected and IDs 2 and 4 not belonging to relation-to-relation NSG disappear. The contiguous links with **PaperTopic** of the complete query graph appear: IDs 3, 5, and 7, the common attribute edge between **PaperTopic** and **Author**. Double-clicking on the last line specifies the natural join of **PaperTopic** and **Author**. The query graph appears in bold on the display.

6 CONCLUSION

The normalized semantic graph of the URI model is a bold attempt to assist the user in database designing and querying: it is automatically constructed and represents much of the database semantics in a condensed and evocative visual form. However, it graphically details only part of the common attributes. It also represents ID in a condensed form (between PURs) but, *a priori*, not in a relation-to-relation representation. It does not use transitive arcs and compatibility edges. For being of the greatest assistance possible to the user in the formulation of a query, so that he only has to select already displayed links, we showed how to obtain the complete query graph made up of all the possible join links between tables. Yet the URITalk interface remains user-friendly despite the large number of possible joins because it displays them only when necessary, at the request of the user.

Further work is necessary

- to see if the relation-to-relation NSG cannot be enriched and made to draw more from the NSG without losing its evocativeness, if the complete query graph cannot shed some of its

transitive arcs without losing its power or if it can be partially implemented by extended inheritance,

- to validate the usability of URITalk interface such as [Cartaci 95],
- to extend to more complex queries [Klein 95] but always constructed from the normalized semantic graph.

In conclusion, we would like to emphasize that an interface such as the one proposed in the present paper is also a prototyping tool. It also can be used to compare the hypertext approach [Conklin 87] [Nielsen 90] with declarative approaches in matters of database querying as well as to enhance database administration.

We thank Claude Boksenbaum, Thérèse Libourel and Uzoma Chukwu for their attentive comments.

7 REFERENCES

Andonoff E., Mendiboure C., Morin C., Rougier V., Zurfluh G. (1992) Une interface graphique pour l'interrogation d'une base de données orientée objet. *VIIIème Journées Bases de Données Avancées*, Tregastel, INRIA Ed, 258-276.

Andraws A., Pichat E. (1991) EAPlus ou l'introduction de rôles dans le modèle Entité-Association. *Journées Bases de Données Avancées*, Lyon, INRIA Ed, 93-111.

Atzeni P., Chan E. P. F. (1987) Independent database schemes under functional and inclusion dependencies. *Proceedings of the 13th VLDB Conference*, Brighton, 159-166.

Auddino A., Dennebouy Y., Dupont Y., Fontana E., Spaccapietra S., Tari Z. (1992) SUPER : A Comprehensive Approach to DBMS Visual User Interfaces. *Actes du 1er séminaire sur les BD 91*, Alger, 341-364.

Cartaci T., Santucci G. (1995) Diagrammatic Vs Textual Query Languages: A Comparative Experiment, in *Proceedings of the 3rd IFIP 2.6 Conference on Visual Database Systems* (ed. S. Spaccapietra and R. Jain), Chapman & Hall.

Casanova M. A., Fagin R., Papadimitriou C. H. (1984) Inclusion dependencies and their interaction with functional dependencies. *Journal of Computer and System Sciences*, **28**, 29-59.

Codd E. F. (1970) A relational model of data for large shared data banks. *Comm. ACM*, **13**, 6, 377-387.

Conklin J. (1987) Hypertext : An Introduction and Survey. *Computer*, September, 16-41.

Fallouh F., Pichat E., Saker D. (1991) Le contrôle de l'intégrité référentielle et de dépendances d'inclusion. *Actes de congrés INFORSID*, Paris, 273-291.

Fallouh F. (1994) Données complexes et Relation Universelle avec Inclusions : une aide à la conception et à l'interrogation des bases de données. *Thèse de Doctorat*, Université Lyon 1.

Goldman K.J., Goldman S.A., Kanellakis P.C., Zdonik S.B. (1985) ISIS: Interface for a Semantic Information System. *ACM*, 0-89791, 328-342.

Heuser C.A., Peres E.M., Richter G. (1993) Towards a complete conceptual model: Petri nets and Entity-Relationship diagrams. *Information Systems*, **18**, 5, 275-298.

IBM (1981) SQL/Data System: Concepts and Facilities. GH-24-5013-0, IBM corporation, White Plains, NY.

Kari S., Kangassalo H., Pösö J. (1990) Conceptual Query Language - CQL: a visual user interface to application data bases. IOS Press, Amsterdam, 608-623.

Klein H.- J., Krämer D. (1995) NQS - A Graphical Query System for Data Models with Binary Relationship Types, in *Proceedings of the 3rd IFIP 2.6 Conference on Visual Database Systems* (ed. S. Spaccapietra and R. Jain), Chapman & Hall.

Kuntz M. (1990) Description et évaluation de Pasta-3, une interface graphique manipulation directe aux bases de données avancées. *Journées Bases de Données Avancées*, Montpellier, INRIA Ed, 123-140.

Lafaye M.-C. (1982) Outil d'aide à la conception des bases de données relationnelles ou réseau (Multigraphe de projection). *Thèse de 3ème cycle Informatique*, Université de Rennes I.

Lakhal L. (1986) Contribution à l'étude des interfaces pour non-informaticien dans la manipulation directe aux bases de données avancées. *Thèse de doctorat*, Université de Nice.

Maier D. (1983) The theory of Relational Databases. Computer Science Press, Potomac, Md.

Mannila H., Räihä K.-J. (1986) Inclusion dependencies in database design. *Int. Conf. In Data Engineering*, Los Angeles, 713-718.

McDonald N.H. (1975) CUPID : a graphics oriented facility for support of non-programmer interactions with a data base. Memorandum M563, University of California, Berkeley.

Melton J., Simon A. (1993) Understanding the new SQL; A complete guide. Morgan Kaufmann.

Mitchell J.C. (1983) The Implication problem for functional and inclusion dependencies. *Information and control*, **56**, 154-173.

Nanard J. (1990) La manipulation directe en interface homme-machine. *Thèse de doctorat d'Etat*, Université Sciences et Techniques du Languedoc, Montpellier.

Nielsen J. (1990) The art of navigating through HYPERTEXT. *Communications of the ACM*, **33**, 3, 297-310.

Olle T.W., Sol H.G., Verrijn-Stuart A.A. (ed.) (1982) Information systems design methodologies. A comparative Review. *Proceedings of the IFIP WG 8.1 Working Conference*, Noordwijkirhout, the Netherlands, 10-14 May, North-Holland.

Pichat E., Bodin R. (1990) Ingénierie des données. Masson.

Pichat E., Roche A. (1989) Extension de la normalisation aux dépendances d'inclusions. *Actes du Congrès INFORSID 89*, Nancy, 57-77.

Ramos H. (1991) IQL : Interface Graphique Paramétrable pour les requêtes relationnelles. *Actes du 1er séminaire sur les BD 91*, Alger, 365-385.

Jeulin P. (1988) GraphTalk : environnement objets de développement et d'utilisation d'ateliers de génie logiciel. Rank Xerox France, 1988.

Reisner P. (1981) Human factors studies of DataBase Query Languages, a survey and assessment. *Computing Surveys*, **13**, 1, 1981.

Roche A. (1990) Définition et réalisation d'un logiciel d'aide à la conception de systemes d'information. *Thèse de Doctorat*, Université de Montpellier II.

Saker D. (1993) L'interrogation des bases de données relationnelles assistée par le Graphe Sémantique Normalisé. *Thèse de Doctorat*, Université Lyon 1.

Schneiderman B. (1983) Direct manipulation, a step beyond programming language. *IEEE computer*, **16**, 8.

Trepied C., Schneider M. (1989) Candid : un modèle sémantique et un langage d'interrogation gaphique pour l'utilisateur final. *INFORSID 89*, Nancy, 183-201.

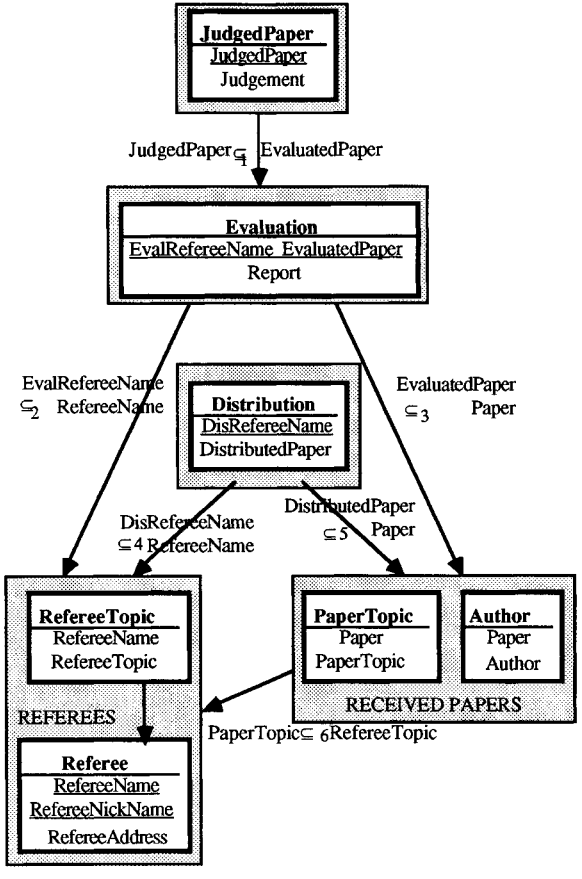
Ullman J. D. (1989) Principles of Database and Knowledge-Base Systems. Volume I & II, Computer Science Press.

Wong E. , Youssefi K. (1976) Decomposition strategy for query processing. *ACM Trans. on Database Systems*, **1**, 3, 223-241.

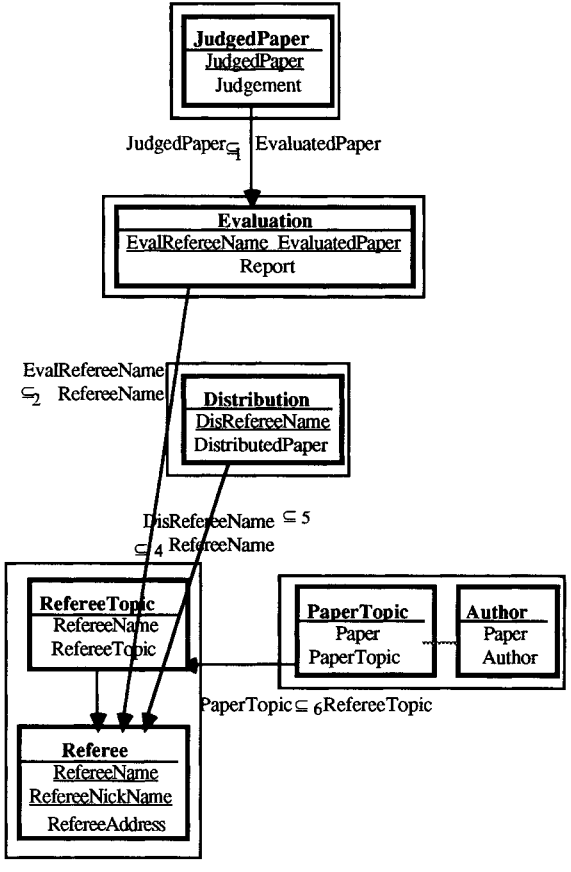
Wu C. T. (1986) A New Graphics User Interface for Accessing a Database. *Proceedings Conference on Advanced Computer Graphics*, Tokyo, Springer Verlag, 203-218.

Wu C. T., Hsiao D. K. (1989) Implementation of visual database interface using an object-oriented langage. *Visual Database Systems*, T. L. Kunii (Editor), Elsevier, IFIP, 105-125.

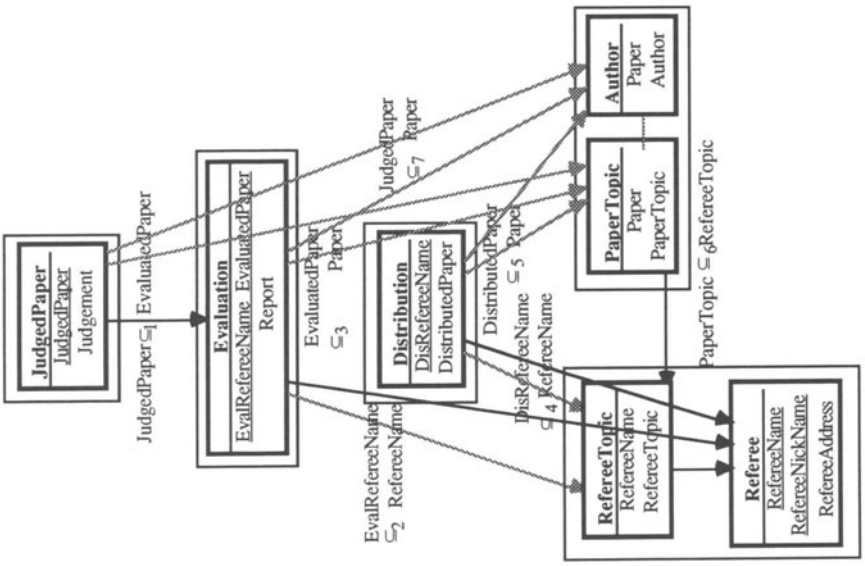
Zloof M. (1977) Query By Example : a database language. *IBM systems journal*, **16**, 1977, 324-343.



Appendix 1 Normalized Semantic Graph.

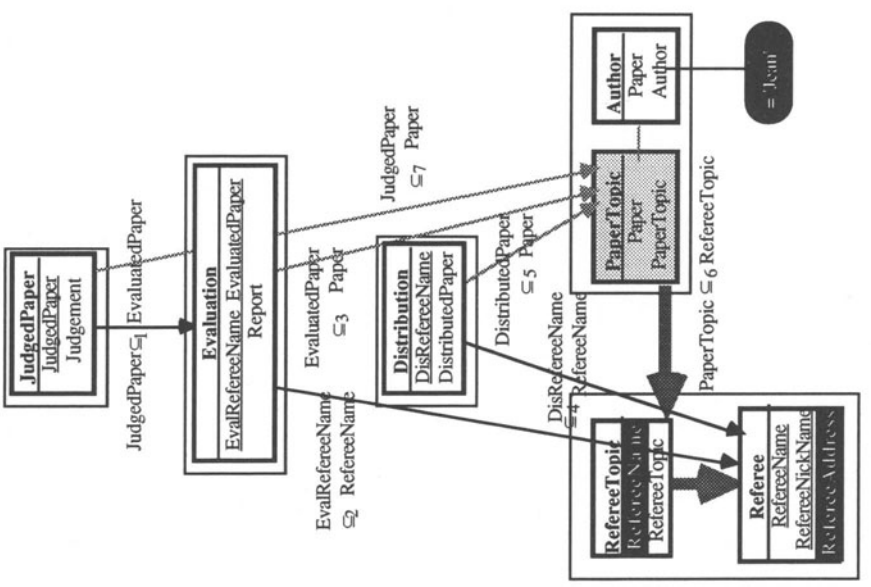


Appendix 2 Relation-to-relation NSG.

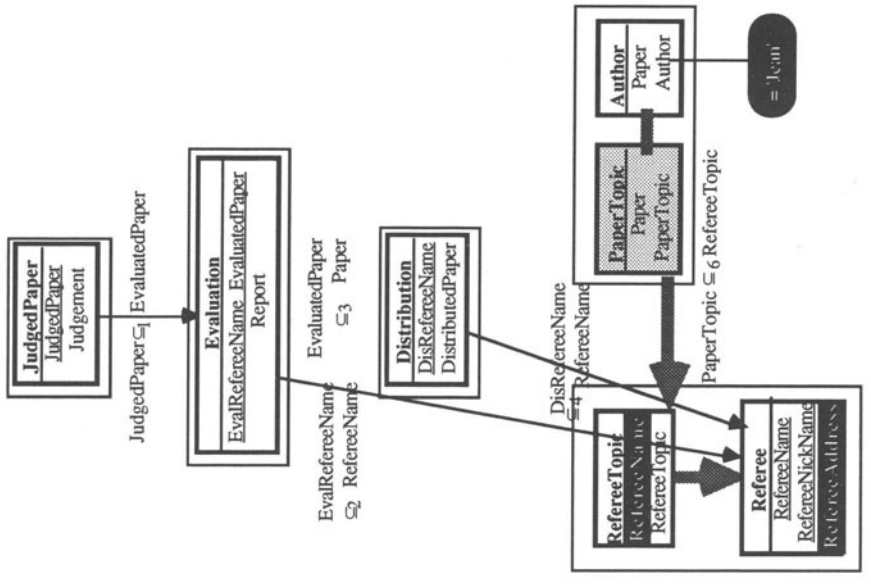


Appendix 3 Complete query graph (relation-to-relation NSG included).

Appendix 4 Evolving query graph with the selected node **RefereeTopic.**



Appendix 5 Evolving query graph with the selected node **PaperTopic.**



Appendix 6 Query graph (and relation-to-relation NSG).

Etienne Pichat

Etienne Pichat received the Engineer Diploma in 1963 from the Ecole Centrale des Arts et Manufactures, Paris and the D. Sc Degree in 1970 from the IMAG Institute, University of Grenoble. He is currently a professor of Computer Science at Lyon 1-Claude Bernard University. He has published papers and books in finite order sets, databases and data models.

Dib Saker

Dib Saker received the Engineer Diploma in 1987 from ISSAT, Damas and the Ph.D. degree in Computer Science from Lyon 1 University in 1992. Currently he is a member of the technical staff at Nat Systems, Paris. His main research interests are in database interfaces and distributed databases.