To Table or Not to Table: a Hypertabular Answer

Giuseppe Santucci

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113, I-00185 Roma, Italy
santucci@infokit.dis.uniroma1.it

Laura Tarantino

Dipartimento di Ingegneria Elettrica
Università degli Studi dell'Aquila
Poggio di Roio, I-67040 L'Aquila, Italy
tarantino@vaxaq.cc.univaq.it

Abstract

Suitable data set organizers are necessary to help users assimilating information retrieved from a database. In this paper we present (1) a general hypertextual framework for the interaction with tables, and (2) a specialization of the framework in order to present in hypertextual format the results of queries expressed in terms of a visual semantic query language.

1 Introduction

Pure relational databases are widely diffused and accessed by diverse classes of users. Much work has been done for devising user interfaces able to significantly reduce the user's effort in interacting with Data Base Management Systems (DBMSs) for selecting, digesting and assimilating information. Notwithstanding that the ultimate goal of the information consumer is the result of the seeking process and not the query formulation per se, most DBMS front-ends provide effective (visual) support for the information *selection* while lacking adequate support for information *digestion* and *assimilation* (with notable exceptions, as [1]).

Suitable data set organizers are necessary to help users make sense of retrieved information. Organizers have to make patterns visible, capture relevant regularities, and allow the construction of new information patterns from old [3]. A static and a dynamic aspect can be singled out in such tools: the *static* aspect refers to the visualization techniques, while the *dynamic* aspect refers to the modalities offered to interact with the visual structures.

Tables are widely used for the visualization of relational query results for their effectiveness in the analysis of structured alphanumeric data sets: they are a very familiar data organization, and require low cost graphical representations. Tables may suffer from a number of problems that, if not adequately addressed, sensibly decrease their efficacy. A first kind of problems is application dependent and relate to the content of tables as query results, while a second type is application independent and pertains to the mapping from logical tables to graphical windows. In the remainder of this section, we discuss these two classes of problems along with possible solutions for overcoming them by acting mainly on the dynamic aspects, in contrast with other approaches that are mainly focussed on the static ones (e.g., [5, 6]).

With regards to the first class of problems, we observe that user requests often require the join of two or more relational tables, which, in general, may result in a table that is not in third normal form, thus exhibiting a tedious repetitions of values. The readability of the table is also decreased by the contiguity of unrelated data, such as attribute values gathered from different tables whose coupling may be meaningless. We observe that when the DBMS is equipped with a query interface based on a semantic model, additional information stored into the semantic representation of data can be used to define output presentations richer than flat sets of tuples.

As to the mapping from logical tables to graphical windows, the approach used when the table dimension exceeds the dimension of the window is to regard the window as a view panning over the table. This is equivalent to having a continuum of adjacent sub-tables, successively disclosed by means of horizontal and vertical scrolling steps (by acting on scroll bars). The main drawback is the fact that a view may clip out relevant portion of the structure (e.g., attributes that identify the rest of the values), giving raise to not meaningful sub-tables. The cause for this undesired *loss-of-context* effect can be found in the nature of the interaction provided by the scroll bars, purely syntactic because originated in the windowing system and not in the specific application[1]. The application must hence have as much control as possible over the interaction, without relying on system-specific tools.

We also recall that cognitive studies on *display density* show that it is not effective to force too much information into one (overloaded) display. It is preferable to map the total volume of information onto a set of smaller displays,

[1] Some tools (e.g., spreadsheets) provide a partial solution to this problem by allowing to split the sets of columns and rows into independent scrollable subgroups, leaving on the user the burden of identifying the portions of the table that have to remain displayed for obtaining meaningful sub-tables.

each containing a closely related subset of information, between which the user should easily move using navigation techniques. In our framework this leads to the association of one (large) relation with a sets of linked display.

In this paper we present our approach for solving the above discussed problems that, being different in nature, have to be faced separately. In Section 2, to suitably handle the mapping from logical tables to graphical windows, we introduce a general interaction structure, called *hypertable*, based on the assumption that the information should be presented on demand as a set of interconnected displays. The displays are dynamically generated on the basis of window dimensions, metadata information, and suitable exploration paradigms somehow captured by the interdisplay links. The hypertable is general in the sense that it does not assume any particular exploration strategy. Then, in Section 3, we specialize the approach to a semantic visual query language, to manage the query result through the *Table Expander*, a tool able to arrange the output of the query in a hypertabular manner, on the basis of the cardinalities of the involved relationships .

2 The interaction model

Interaction models are introduced to bridge the conceptual distance between the visualization model and the data model, and to provide the structures for the user interaction. In this section we outline a methodology for defining interaction models based on *hypertables*, which are multi-display visualizations of (large) relations embedding links among relation fragments. As the output device is limited in terms of the containment area, the visual and the interaction models must be designed in terms of representations consequently constrained. We hence introduce the concept of *bounded relation* (or *fragment*), with upper-bounds on the number of tuples and on the number of schema attributes. Roughly speaking, any relation r is represented by a set of fragments from which it is possible to retrieve all the information contained in r. More formally:

Definition 1 Let U be a universe of *attributes*, and let $dom(A)$ be the *domain* of *values* associated to each attribute A in U. A *relation schema* R is a non empty subset of U. A *tuple* t on R is a function that associates a value $t(A)$ in $dom(A)$ to each attribute A in R. A *relation* $r(R)$ is a finite set of tuples on R. We say that $r(R)$ is an *(h,w)-relation* on R if w is the cardinality of R and h is the number of tuples of r. We refer to the ordered pair (h, w) as to the *dimensions* (*height* and *width*) of r.

When either dimension of a relation r leads to a table exceeding the display dimensions, it is necessary to associate to r a set of (h, w)–relations (*fragments* of r), where both h and w satisfies the dimensional constraints of the display.

Definition 2 A set F of fragments is said an *F-representation* of a relation r if and only if there exist two computable procedures *split* and *coalesce* such that $F = split(r, h, w)$ and $r = coalesce(F)$.

This definition of *F-representation*, though similar to the definition of lossless decomposition typical of the normalization process of the relational theory, differs from it in two ways: (1) dimensional constraints are used to define the fragments, (2) we do not limit the set of attributes and values in F to be subsets of those belonging to r (e.g., fragments containing aggregated values may belong to an F-representation). A consequence of the second aspect is that, in principle, there are infinite F-representations associated to r. It is therefore necessary to enforce some properties that F has to satisfy in order for the representation to be *good* with respect to the interaction.

Non redundancy We say that F is *non redundant* when no proper subset of F is an F-representation of r. Among non-redundant representations, it might be preferable to select those with *minimal* size. From the interaction point of view non-redundancy and minimality are important for simplifying the navigation process needed to visit F.

Meaningfulness Criteria for the meaningfulness may include: short (semantic) distances among fragments and between the fragments and the original relation, the existence of a *canonical kernel* of fragments, e.g., including *starting points* for the navigation carrying some sort of summarized information and/or hints for the navigation.

The two aspects are not independent, and meaningfulness plays a prioritary role.

While fragments provide the navigation space, an adjacency structure linking them defines the admissible interaction. Depending upon the paradigm underlying the *split* and *coalesce* procedures, the links might be attached to the fragment as a whole, or to some element of it (e.g., attributes, values or tuples). To be *good* with respect to the interaction, the adjacency structure must satisfy a number of criteria (we refer to [9] for an extensive discussion).

Completeness The set of links must guarantee that each fragment is reachable through interaction paths originating in the kernel.

Regularity Some sort of consistency must be given to the adjacency structure, to enhance the navigation: the predictability of regular patterns allows the user to scan more easily the area of interest (introducing also significant aesthetic benefits).

Meaningfulness Intuitive semantics must be defined for the links, according to navigation paradigm underlying the *split* procedure.

Meaningfulness criteria are introduced both in the F-representation and in the adjacency structure to ensure effective interaction. *Orderings* on fragment schemata and on fragment extensions may also be required to help the navigation (e.g., it may be useful to visualize related attributes in adjacent columns of a table). As a matter of fact, ordering is somehow implied also by the existence of oriented links among fragments. To reflect these requirements in the interaction model, additional concepts are introduced.

Definition 3 An *(h-w)-table* is a triple $\langle r, S1, S2 \rangle$ where r is an (h, w)-relation on a schema R, $S1$ is a sorting of R, and $S2$ is a sorting of the tuples of r. $S1$ and $S2$ may remain unspecified when no predefined ordering is required (we will refer to (h-w)-tables simply as tables).

Definition 4 Let T be a set of (h, w)-tables. The set $F = \{f \mid \langle f, s1, s2 \rangle \in T\}$ is said *induced* by T.

Definition 5 An *hypertable* HT is an ordered pair $\langle T, L \rangle$, where T is a set of (h, w)-tables, and L is an adjacency structure defined over the set of fragments induced by T.

To ensure effective interaction, the above discussed properties must be enforced on the hypertable components T and L.

3 The Table Expander

In this section we show how the approach can be specialized to a given query interface and query result exploration paradigm. In particular, we focus on a visual semantic query language, and, after a brief description of the query formulation strategy, we show how the information stored into the semantic schema can be usefully exploited for (1) fragmenting the resulting table, and (2) interconnecting such fragments in hypertabular manner.

The Query Formulation The availability of a high level description of the database through a semantic model (e.g., the ER model [4]) results naturally in query interfaces in which the user visually interacts with a diagram representing the underlying semantic schema. The query formulation is based on a *navigational* approach (see [2]), in which the user specifies a path among the classes and the relationships of the schema, like the one shown in Fig. 1. The path corresponds to an ordered sequence of joins between pairs (entity, relationship), followed by a final selection and projection (a general discussion on the matter can be found in [7]).

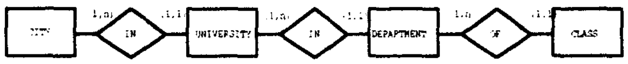
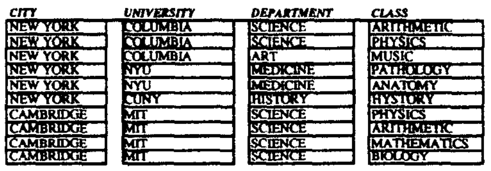


Figure 1. *A sample query path*

Isolating entity attributes To introduce our query result exploration paradigm we use the example of Fig. 1. The resulting table contains attributes coming from different entities, and may present many repetitions of values.

The first natural choice in the fragmentation process is the isolation of clusters of attributes belonging to the same entity. Since an entity represents a class of real world objects sharing common properties, its common attributes possess a strong cohesion. Furthermore, at least under the hypothesis that the ER schema satisfies the first normal form requirements, these attributes are all in a one-to-one relationship with one another.[2] Having isolated the entity attributes, we leave in the original table one attribute for entity, typically a meaningful key. Each attribute will be the source of a link pointing to a fragment whose schema is the associated attribute cluster. The new (restricted) table $\mathcal{T}$ will be like the one in Fig. 2, containing many repetitions of values. One may notice that $\mathcal{T}$ can be iteratively partitioned from left to right, on the basis of repeated values. This is due to the particular orderings chosen for placing the attributes: the attributes in a one-to-many relationships with all the others (i.e., showing the greatest number of repetitions) appear first on the left, and the more we go to the right, the greater is the number of distinct values appearing in a colums. This is exactly the way a table is expected to be: more general to the left, more specific to the right.[3]



CITY	UNIVERSITY	DEPARTMENT	CLASS
NEW YORK	COLUMBIA	SCIENCE	ARITHMETIC
NEW YORK	COLUMBIA	SCIENCE	PHYSICS
NEW YORK	COLUMBIA	ART	MUSIC
NEW YORK	NYU	MEDICINE	PATHOLOGY
NEW YORK	NYU	MEDICINE	ANATOMY
NEW YORK	CUNY	HISTORY	HYSTORY
CAMBRIDGE	MIT	SCIENCE	PHYSICS
CAMBRIDGE	MIT	SCIENCE	ARITHMETIC
CAMBRIDGE	MIT	SCIENCE	MATHEMATICS
CAMBRIDGE	MIT	SCIENCE	BIOLOGY

Figure 2. *A reduced table*

The following steps are hence: (1) to devise an algorithm for automatic attribute sorting of the above type, and (2) to take advantage of such sortings for the definition of a proper hypertabular structure.

Sorting the attributes The problem now is to compute the arity of a relationship between two clusters of attributes coming from two different ER concepts. Two cases are given: (1) the attributes under consideration are part of two concepts of the query path directly linked or (2) they are still on the same path, but "far" from each other. In the first case the numerical proportion can be derived directly from the cardinality of the ER schema, the maximum one being an indication of the greatest number of instances of one entity for each instance of the

[2] Should such entity fragments lead to tables exceeding the display dimensions, they can be further fragmented (see [9]).

[3] For non Western cultures the natural visual scanning is instead from right to left.

other. The second case is more complex. Evidence must be found by navigating through the schema from one attribute to the other; cardinalities found through a path must be combined to determine a *composed cardinality* between two objects of the path not directly linked.

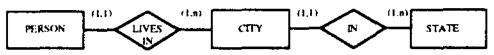
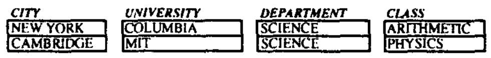


Figure 3. *A three-step path*

Let us briefly describe the case of two near entities in the path of Fig. 3 (that we assume specified from left to right), in which we denote with a pair the minimum and maximum cardinalities of a relationship. Note that cardinalities are considered from the point of view of the relationship itself: the pair (1,1) between *Person* and *Lives – in* means that each instance of *Person* is involved in the relationship *Lives – in* at least and at most once; each instance of *City* is involved in the same relationship at least once and at most n times. Moreover, when comparing two entities we often talk only about the maximum cardinalities and we say, e.g., that *Person* and *City* are in many-to-one relationship through *Lives – in*. Composing the cardinalities, *Person* is found to be in a (many-to-one)^2 relationship with *State*, since *Person* is in many-to-one relationship with *City*, and *City* is in many-to-one relationship with *State*. In this case, the natural presentation of attributes is in contrast to the specified path flow. It is more effective, in fact, to present first the attributes coming from *State*, showing for each state the set of its cities and for each city the set of its inhabitants. If we show the attributes in the order corresponding to the path, there is no way of avoiding the tedious repetition of the city for all the persons living in the same city and the repetition of the state for each city in that state (a deeper description of the strategies for calculating the numerical proportion between two attributes belonging to a complex path can be found in [8]). Summarizing, on the basis of the arity of the relationships of the ER schema, a partial order relation on the table attributes is given. Such a partial order is matched against the total order defined by the user path that, when necessary, is altered according to the above considerations.

The expansion paradigm Once the numerical relationships are defined, they are used to generate meaningful fragments. As discussed before, the distinct values in any column determine a partition of the subtable on the right of the column. Let us consider the first attribute. Two distinct values appear, namely *NewYork* and *Cambridge*. It makes sense to provide these two values as starting hints for the exploration of the query result. The user can then select one of the two to see the associated information. To enforce regularity, and let the user feel completely free to browse through the entire table, the remaining columns are maintained. Therefore a reasonable *starting fragment* for the interaction is the table $\mathcal{T}_1$ shown in Fig. 4, in which only the distinct instances of the first column appear, each completed with one of the tuples associated to it (typically the first one).



CITY	UNIVERSITY	DEPARTMENT	CLASS
NEW YORK	COLUMBIA	SCIENCE	ARITHMETIC
CAMBRIDGE	MIT	SCIENCE	PHYSICS

Figure 4. *The starting fragment*

At this point, a method must be introduced to allow the selective presentation of new data. For each distinct instance of the first column three additional fragments are defined and linked to it, with schemata $\langle University, Department, Class\rangle$, $\langle Department, Class\rangle$, and $\langle Class\rangle$, respectively, and with extensions defined following an approach consistent to the one used for building $\mathcal{T}_1$ (i.e., a single tuple is included for each distinct instance). From the user point of view, these new tables can be thought of *expansions* of $\mathcal{T}_1$. To expand $\mathcal{T}_1$, the user must select one instance of *City* and one attribute on its right (thus uniquely identifying one of the three links attached to the instance value). Fig. 5 shows the selection of *New York* and its further expansion starting from *University*, while Fig. 6 shows the expansion of *New York* starting from *Department*.

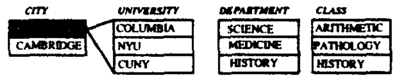


Figure 5. *Expansion of the value New York*

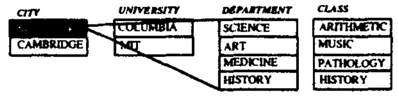


Figure 6. *A different expansion of New York*

It should be intuitive that such an approach can be applied to any attribute of $\mathcal{T}_1$ as well as to any attribute of any new fragment. Selection and expansion can involve almost every pair of columns, with the only constraint that the expansion is performed on the right of the selection, since everything that is on the left acts as a "key" to the sub-table[4]. By specifying the attribute to be expanded, a new fragment is obtained (whose schema contains the expanded attribute and the columns on its right),

[4]For this reason we do not fix a maximum arity for fragment schemata (during horizontal browsing, in fact, any table is always identified by its leftmost column). Furthermore it is unlikely the necessity of handling very long paths involving semantically distant concepts.

which follows the same rules of the initial one: a single tuple for each distinct instance of the leftmost attribute of the fragment. When the attributes involved in the selection/expansion specification are not adjacent (this is the case shown in Fig.6), the columns lying between them are not expanded, for two reasons: (1) no operation was required on them; (2) repetitions would be introduced, worsening readability . This solution also helps multiple expansions based on a single selection. As an example, in Fig. 7 multiple expansions are shown (the figure comes from the current implementation of the prototype).

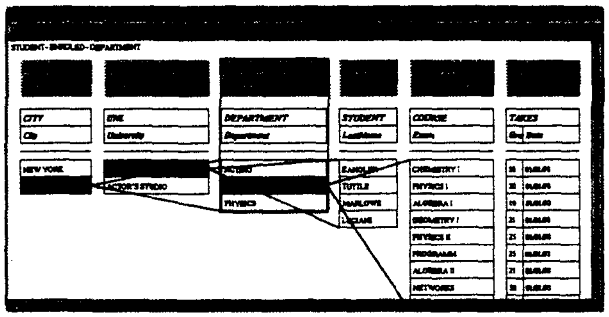


Figure 7. *Multiple expansions*

The prototype A prototype of the system has been implemented under the Unix operative system, using C++ language and the XVT graphical toolkit. This toolkit is a set of libraries available for different platforms (Dos, Mac-OS, and Unix) and ensures portability over several environments (hardware and software). Some additional features have been included in the prototype to enrich the interaction. A *change-selection* method is supplied. It is likely that the user is interested in moving through the instances of one attribute to browse different sub-table. For example, after the analysis of the information related to $NewYork$ in Fig. 5, the selection of $Cambridge$ yields the table shown in Fig. 8.

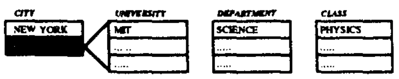


Figure 8. *Expansion of the value Cambridge*

To the left of the selection, however, there might be many expansions depending on it, directly or indirectly. The default approach is to contract all subsequent expansions, related to the changed selections, forcing them to fit the expansion level of the column containing the selection. A *keep-expansions* option, anyway, allows the user to maintain the expansion pattern. Finally, the user is always able to bring things back to a situation equal – or at least compatible if something else has changed – to the one present before an expansion (proper strategies are supplied for propagating such *contractions*).

4 Conclusion

In this paper we presented our approach, and an application based on it, for overcoming typical problems encountered by users when interacting with tables. The main idea is based on the assumption that the information is presented on demand as a set of interconnected displays, dynamically generated by the system. Given its hypertextual nature, the proposed interaction structure is also suitable as a methodological framework for the definition of user interfaces in innovative environments (as the World Wide Web).

References

[1] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *CHI '94*, pages 151–164, ACM, New York, 1994.

[2] M. Angelaccio, T. Catarci, and G. Santucci. Qbd*: a graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150–1163, 1990.

[3] S.K. Card. Visualizing retrieved information: a survey. *IEEE Computer Graphics and Applications*, 16(2):63–67, 1996.

[4] P.P. Chen. The entity relationship model toward a unified view of data. *ACM Transactions on Data Base Systems*, 1(1), 1976.

[5] R. Chimera. Value bars : an information visualization and navigation tool for multi-attribute listings. In *CHI '92*, pages 293–294, 1992.

[6] R. Rao and S.K. Card. The table lens: Merging graphical and symbolic representation in an interactive focus + context visualization for tabular information. In *CHI 94*, pages 318–322, 1994.

[7] G. Santucci. On graph-based interaction for semantic query languages. In *Proc. of IEEE 1996 Symposium on Visual Languages*, pages 76–83, 1996.

[8] G. Santucci and F. Palmisano. A dynamic form-based data visualizer for semantic query languages. In *Sawyer P (ed) Interfaces to Database Systems*, pages 249–265, 1994.

[9] L. Tarantino. Hypertabular representations of database relations in world wide web front-ends. In *(Barclay, P. and Kennedy, J. eds) Interfaces to Databases (IDS-3)*, Springer Verlag, 1996.