Interactive Query Formulation using Query By Navigation
Confidential

Asymetrix Report 94-4

H.A. Proper
Asymetrix Research Laboratory
Department of Computer Science
University of Queensland
Australia 4072
E.Proper@acm.org

Version of June 23, 2004 at 10:29

PUBLISHED AS:

H.A. Proper. Interactive query formulation using query by navigation. Asymetrix Research Report 94-4, Asymetrix Research Laboratory, University of Queensland, Brisbane, Australia, 1994.

Abstract

Effective information disclosure in the context of databases with a large conceptual schema is known to be a non-trivial problem. In particular the formulation of ad-hoc queries is a major problem in such contexts. Existing approaches for tackling this problem include graphical query interfaces, query by navigation, query by construction, and point to point queries. In this report we propose an adoption of the query by navigation mechanism that is especially geared towards the InfoAssistant product. Query by navigation is based on ideas from the information retrieval world, in particular on the stratified hypermedia architecture.

When using our approach to the formulations of queries, a user will first formulate a number of simple queries corresponding to linear paths through the information structure. The formulation of the linear paths is the result of the *explorative phase* of the query formulation. Once users have specified a number of these linear paths, they may combine them to form more complex queries. Examples of such combinations are: concatenation, union, intersection and selection. This last process is referred to as *query by construction*, and is the *constructive phase* of the query formulation process.

1 Introduction

This report is concerned with an adaption of the existing idea of query by navigation to make it fit within the InfoAssistant product. Query by navigation has been discussed before in [BPW93], [PW95], [Pro94a], and [HPW96]. In this report we base ourselves mainly on the latest version of the query by navigation mechanism as discussed in [HPW96]. We have, however, stripped the existing definition from its information retrieval context to tailor it better to the InfoAssistant context.

In the previous Asymetrix reports [Pro94b] and [Pro94c], we have already provided an elaborate motivation for the query formulation tools we envision for InfoAssistant. Therefore, we do not provide any further motivation for query by navigation in this report and limit ourselves to the discussion of the idea itself and its formal background.

The structure of this report is as follows. In section 2 a sample query by navigation session is discussed, while a discussion of the foundations in terms of ORM and path expressions is provided in section 3. The core of this report is formed by section 4, in which the actual query by navigation graph is defined. Finally, section 5 concludes the report. For the reader who is unfamiliar with the notation style used in this report, it is advisable to first read [Pro94d].

2 Exploring an Information Structure

Before formally describing the notion of query by navigation, we offer the reader a brief example of the use of a query by navigation system. The example shows how the system can support users when they formulate a query. In our view, the process of query formulation corresponds to a search through the information system with the aim to gradually fulfill a user's information need. Query by navigation is one of the avenues along which parts of this information need can be formulated. These partial formulations can then be used in the constructive phase of the query formulation where they are integrated into a complete query.

During query by navigation, the (partial) query of the searcher is formulated by stepwise refining or enlarging the current description (the *focus*) of this query, until the searcher recognises the current description as the best possible description of this (part of the) query. In the example we make use of the conceptual schema of the presidential database as depicted in figure 1.

The first node shown to the user is depicted in figure 2. In the upper window the query by navigation dialogue is shown, whereas the lower window displays the current query by construction session. The query by navigation window displays the standard starting node of a query by navigation session; it simply lists all object types in the conceptual schema. In most Object-Role Modelling dialects and ER variations, relationship types can be objectified, i.e. instances of relationship types can play roles in other relationship types. In the query by navigation system, relationship types are not treated as object types until they indeed have been objectified. So non-objectified relationship types are not listed in the start window.

In the context of query by navigation, we use the term *node* to refer to the screens shown to the user in the query by navigation window. This is done to emphasize the fact that the navigation during a query by navigation session can be seen as the navigation through a graph (or a hypertext).

Each entry in a node represents one way to continue the search through the conceptual schema. A node thus corresponds to a moment of choice in the search process. The order in which the alternatives are listed in the starting node, and nodes in general, can be based on multiple factors. In this paper we do not discuss these factors in detail, but alternatives may be based on the conceptual relevance of the object types occuring in the alternatives ([CH94], [Pro94b]), or the user's past behaviour ([BHW96]).

Let us presume the user is interested in presidents who are married and the number of children that resulted from these marriages. In the starting node, the user may select 'the president' as the first refinement of the information need. This leads to the example node as presented in figure 3. The associated node shows the direct environment of type president. This new node contains three classes of entries. Firstly, the ⬆ button takes the user back to a more general node, in this case the starting node.

The other class (⬇ button) represents the possible refinements of the current focus. This set basically consists of the following two classes:

1. For each n-ary relationship type in which the current focus (the president) plays a role, we have $n-1$ possible refinements since there are $n-1$ possible ways to continue the path *through* this relationship type.
2. Each role leading into an objectified relationship type (e.g. Marriage), or a non-binary relationship type, also results in a possible refinement.

The second class is needed to cater for the traversal of objectified relationship types (in the direction of the objectification), and furthermore, to be able to split paths on n-ary relationship types (e.g. into $n-1$ paths).

The last group of entries in the node (the ➡ buttons) are the associative links. They are derived from the subtyping hierarchy in the conceptual schema. In our example ORM schema these are the supertypes of the President object type, being Politician and Person.

The searcher selects the president who is involved in a marriage as the next focus, i.e. the objectified relationship type is the next point for possible further refinement. In the resulting node the searcher continues with president involved in marriage with person.

The user then decides to select the refinement with a person. So effectively, the user has traversed an (objectified) binary relationship type in two steps. (Note that the system verbalised this traversal in a briefer format). The first step brought the user to the Marriage relationship type, while the second step brought the user to the Person object type. This leads to the node depicted in figure 5. Note that the user could just as well have selected the president who has as spouse a person in the node depicted in figure 3.

The user decides that the current focus is, for the moment, a proper description of the information need. To get an impression of query the result so far the user can select the GO! button. This should result in a window showing the result of the path formulated in the query by navigation session.

In [BPW93], [PW95], [Pro94a], and [HPW96], the query by navigation system also included the population. In such a system, a user can either navigate through the population of the schema, or through the conceptual schema (the type level) as discussed in this section. When the query by navigation mechanism, restricted to the type level as we propose for InfoAssistant, turns out to be successful we can indeed consider introducing query by navigation on a population level as well. We realise that the navigation through the population for the formulation of queries is only useful in the context of small database populations. However, navigation through a hyperbase has more uses. Firstly, it can be used as a mechanism for relevance feedback during the query formulation process, since it allows the user to probe the result of the query. Secondly, it is quite possible to use navigation through the hyperbase as a mechanism for the verification of conceptual schemas. In the design procedure for Object Role Modelling techniques example populations play an important role. If a CASE tool allows for the storage of a complete example population, navigating through this population might turn out to be a good tool for the validation by the future users of the conceptual schema.

3 The Basics

To formally define the query by navigation mechanism, we first have to formally define what an ORM schema is and, even more importantly, the (linear) path expressions need to be introduced as they form the backbone of the query by navigation system. We base ourselves on the formalisation of the path expressions as provided in [HPW93], and the formalisation of ORM given in [HP95].

3.1 ORM Schemas

A conceptual schema is presumed to consist of a set of types $\mathcal{TP}$. Within this set of types two subsets can be distinguished: the relationship types $\mathcal{RL}$, and the object types $\mathcal{OB}$. Furthermore, let $\mathcal{RO}$ be the set of roles in the conceptual schema. The fabric of the conceptual schema is then captured by two functions and two predicates. The set of roles associated to a relationship type are provided by the partition: $\mathsf{Roles} : \mathcal{RL} \rightarrow \wp(\mathcal{RO})$. Using this partition, we can define the function Rel which returns for each role the relationship type in which it is involved: $\mathsf{Rel}(r) = f \iff r \in \mathsf{Roles}(f)$. Every role has an object type at its base called the player of the role, which is provided by the function: $\mathsf{Player} : \mathcal{RO} \rightarrow \mathcal{TP}$. Subtyping and polymorphy of object types are captured by the predicates $\mathsf{SpecOf} \subseteq \mathcal{OB} \times \mathcal{OB}$ and $\mathsf{HasMorph} \subseteq \mathcal{OB} \times \mathcal{OB}$ respectively. From SpecOf and $\mathsf{HasMorph}$ we can derive the more general IdfBy relation, capturing inheritance of properties between object types, by the following derivation rules:

1. $x \, \mathsf{SpecOf} \, y \ \vdash \ x \, \mathsf{IdfBy} \, y$
2. $x \, \mathsf{HasMorph} \, y \ \vdash \ x \, \mathsf{IdfBy} \, y$
3. $x \, \mathsf{IdfBy} \, y \, \mathsf{IdfBy} \, z \ \vdash \ x \, \mathsf{IdfBy} \, z$

Using IdfBy we can define the notion of type relatedness: $x \sim y$ for object types x and y. This notion captures the intuition that two object types may share instances. This relation is defined by the following four derivation rules:

1. $x \in \mathcal{TP} \ \vdash \ x \sim x$
2. $x \, \mathsf{IdfBy} \, y \ \vdash \ x \sim y$
3. $x \sim y \ \vdash \ y \sim x$
4. $x \sim y \sim z \ \vdash \ x \sim z$

Note that when using ORM with advanced concepts ([HPW93], [HP95]) such as sequence types, set types, etc., the definition of $\sim$ needs to be refined.

3.2 Linear Path Expressions

The central aspect of query by navigation are the (linear) path expressions ([HPW93]). These expressions are build from (object) types, roles, and instances. For query by navigation in InfoAssistant we only consider (object) types and roles. In a linear path expressions, all these components are interpreted as binary relations, and can as such be concatenated to each other.

A type t occurring in a path expressions corresponds to a binary relationship with tuples $\langle x, x \rangle$ for every instance x of type t. A role r corresponds to a binary (multiset) relationship connecting $\mathsf{Player}(r)$ to $\mathsf{Rel}(r)$, with tuples $\langle x, y \rangle$ where x is the r part of relationship instance y of $\mathsf{Rel}(r)$. To traverse relationship types using a path expression, it must be possible to reverse the order of the $\mathsf{Player}(r)$ and $\mathsf{Rel}(r)$ part of a role. Therefore, $r^{\leftarrow}$ represents the reversed binary relation associated to role r. Note that in a forthcoming Asymetrix research report the path expressions will be discussed in more detail. For the moment, however, refer to [HPW93] for an elaborate formal definition, and to [HPW94] for a more detailed informal discussion.

When displaying linear path expressions in the nodes of the query by navigation mechanism, the linear path expressions need to be verbalised. These verbalisations can be derived from the names given to the types and roles from the conceptual schema. In this report we simply presume the existence of a function ρ verbalising these linear path expressions. For a more detailed discussion on the verbalisation of linear path expressions refer to [Pro94a].

4 The Query By Navigation Graph

In this section we define the query by navigation graph itself. Formally, a query by navigation graph is introduced as a structure $\mathcal{QBN} \triangleq \langle \mathsf{Nodes}, \mathsf{RefineTo}, \mathsf{AssTo} \rangle$. The components of this graph are introduced below. Nodes is the set of nodes of the graph. Two classes of edges for the query by navigation graph are introduced. The first ones ($\mathsf{RefineTo}$) are the ones that are based on the structure of the linear path expressions, whereas AssTo provides the associative connections that are induced by the type relatedness of types.

4.1 Structure based navigation

The set of nodes of the query by navigation graph is defined by means of a set of grammar (context-free production rules). This grammar contains for each type x a corresponding non-terminal (syntactic category) $\langle P_x \rangle$. Instantiations of syntactic category $\langle P_x \rangle$ describe simple properties of (instances of) type x, i.e., properties that can be derived via a linear path expression starting in object type x.

The first rule of the grammar defines what a linear path expression is:

$$\langle PE \rangle \;\rightarrow\; \langle P_x \rangle$$

For any $x \in \mathcal{OB}$ we now have the following rules:

$$\langle P_x \rangle \;\rightarrow\; x$$

The inheritance of properties between types leads to the following rules. If $x \underset{\sim}{\sim} y$, then:

$$\langle P_x \rangle \;\rightarrow\; \langle P_y \rangle$$

which means that properties about y may be used in expressions about x. Each role $r \in \mathcal{RO}$ such that $\mathsf{Rel}(r) \in \mathcal{OB}$ leads to the following rules:

$$\begin{aligned}\left\langle P_{\mathsf{Rel}(r)} \right\rangle &\rightarrow \left\langle P_{\mathsf{Player}(r)} \right\rangle \circ r \circ \mathsf{Rel}(r) \\ \left\langle P_{\mathsf{Player}(r)} \right\rangle &\rightarrow \left\langle P_{\mathsf{Rel}(r)} \right\rangle \circ r^{\leftarrow} \circ \mathsf{Player}(r)\end{aligned}$$

Finally, for all roles r, q such that $r \neq q$ and $\mathsf{Rel}(p) = \{r, q\}$ we have:

$$\left\langle P_{\mathsf{Player}(q)} \right\rangle \rightarrow \left\langle P_{\mathsf{Player}(r)} \right\rangle \circ r \circ \mathsf{Rel}(r) \circ q^{\leftarrow} \circ \mathsf{Player}(q)$$

The set of nodes in the query by navigation graph (Nodes) is now formed by the set of linear path expressions which can be formed by the above rules (starting from $\langle PE \rangle$), augmented with the empty path expression ϵ (which serves as the default starting point of a query by navigation session). Note that the above syntax describes meta-rules, which are concretized by substituting an actual object type for meta-nonterminal x or role q, r. So, basically this grammar is a two level grammar ([WMP$^+$76]).

When navigating through the query by navigation graph, one actually actually navigates by refinement or enlargement of a linear path expression, or by association.

Formally, the refinement/enlargement structure $\mathsf{RefineTo}$ of a query by navigation graph is a subset of $\mathsf{Nodes} \times \mathsf{Nodes}$. This set is identified by the following kinds of structural links:

1. a link from the empty path expression ϵ to any molecule x.
2. a link from a molecule $P\ x$ to a molecule $P\ x \circ r \circ \mathsf{Rel}(r)$ if $x \simeq \mathsf{Player}(r) \wedge \mathsf{Rel}(r) \in \mathcal{OB}$.
3. a link from a molecule $P\ x$ to a molecule $P\ x \circ r^{\leftarrow} \circ \mathsf{Player}(r)$ if $x \simeq \mathsf{Rel}(r) \wedge \mathsf{Rel}(r) \in \mathcal{OB}$.
4. a link from a molecule $P\ x$ to a molecule $P\ x \circ r \circ \mathsf{Rel}(r) \circ q^{\leftarrow} \circ \mathsf{Player}(q)$
   if $r \neq q \wedge \mathsf{Rel}(r) = \{r, q\}$.

where P is any linear path expression, and q, r are roles and x is an object type.

4.2 Association based links

In our running example, each president is also a politician. Sometimes, the user may want to refine a linear path expression ending at $\mathsf{politician}$ to a linear path expression ending at $\mathsf{President}$. This is one of (two) reasons why we introduce the so called associative links. In order to avoid chaotic structures, these links are only included for types occurring at the end of a linear path expression. An example of such a link is a link from the president who has as spouse a person to the president who has as spouse a president and to the president who has as spouse a politician.

The front part of a linear path expressions is manipulated only indirectly when navigating through the graph. To manipulate the front of the path expressions explicitly, path reversal is offered. Path reversal is the second form of associative links. For example, the link from the president who is involved in a marriage to the marriage of a president is a path reversal.

As stated before, we introduce the associative links (AssTo) of the hyperindex to cater for the relations in the identification hierarchy, as well as the reversal of the current focus. Let x, y be object types, then we have the following kinds of associative links:

1. a link from a molecule of the form $P\ x$ to a molecule $P\ y$ if $x \sim y$, capturing type relatedness.
2. a link from molecule P to molecule $\mathsf{Rev}(P)$ if $P \neq \mathsf{Rev}(P)$, catering for the reversal of path expressions.

The reversal of a path expressions by the Rev operation is recursively defined as:

$$\begin{aligned} \mathsf{Rev}(P \circ p \circ x) &\triangleq x \circ p^{\leftarrow} \circ \mathsf{Rev}(P) \\ \mathsf{Rev}(P \circ p^{\leftarrow} \circ x) &\triangleq x \circ p \circ \mathsf{Rev}(P) \\ \mathsf{Rev}(x) &\triangleq x \end{aligned}$$

An example of such a reversal is:

$$\mathsf{Rev}(x \circ p \circ f \circ q^{\leftarrow} \circ y) = y \circ q \circ f \circ p^{\leftarrow} \circ x$$

4.3 Presentation of molecules

Nodes are presented on the screen by verbalising the node itself (i.c. the linear path expression) and all nodes reachable from this node, using AssTo and RefineTo. The set of reachable nodes is called the *direct environment* of the molecule. The presentation of a node is thus made up of:

1. a verbalization of the molecule itself, identifying the current spot (the focus) in the hyperindex.
2. a verbalization of each immediate ancestor, showing how to decompose the focus into its components,
3. a verbalization of each immediate descendant, which suggests how to extend the current focus.

4. a verbalization of each associated molecule, showing the related alternatives.

The presentation of a node n is formally identified as:

$$\mathsf{Present}(n) \quad \triangleq \quad \langle \rho(n), \mathsf{Refine}(n), \mathsf{Enlrge}(n), \mathsf{Assoc}(n) \rangle$$

where the direct environment of M is captured by:

$$\begin{aligned} \mathsf{Refine}(n) &\triangleq \left\{ \rho(r) \;\middle|\; \langle r, n \rangle \in \mathsf{RefineTo} \right\} \\ \mathsf{Enlrge}(n) &\triangleq \left\{ \rho(e) \;\middle|\; \langle n, e \rangle \in \mathsf{RefineTo} \right\} \\ \mathsf{Assoc}(n) &\triangleq \left\{ \rho(a) \;\middle|\; \langle M, a \rangle \in \mathsf{AssTo} \right\} \end{aligned}$$

All that remains to be done with respect to the presentation of the molecules, is a proper definition of $\rho(P)$ where P is a path expression. This can be done by a set of derivation rules, with an associated preference (using penalty points). As stated before, for a more detailed discussion of such a set of verbalisation rules please refer to [Pro94a].

4.4 Navigating through the graph

The navigation through the graph should be clear now. If a user selects an optional refinement/enlargement/association, the associated linear path expression becomes the new focus and the direct environment of this new focus (node) is shown on the screen.

When implementing the query by navigation mechanism, it is probably wise to calculate the direct environment of a node dynamically. This is needed as the number of links in RefineTo and AssTo can, for obvious reasons, be extremely large. The above definitions of RefineTo and AssTo, indeed allow for such a dynamic calculation of the direct environment of a node.

5 Conclusions

In this report we defined a limited version of the query by navigation mechanism that is tailored for the InfoAssistant product. In later versions of InfoAssistant, a more complete implementation of the query by navigation mechanism may be considered. For instance the navigation through the population and the support of multiple abstraction layers ([HPW96]). Furthermore, the verbalisation of (linear) path expressions is an area in which more improvements are possible.

Finally, extensive testing of the user interface of the query by navigation tool is required. A number of possible refinements of the user interface exist. For instance, user might prefer it if refinements based on supertypes or subtypes are explicitly marked as such. For instance:

The president who is (as a person) vice president of an administration instead of

The president who is vice president of an administration

Such a feature could be added to the system as a *user selectable* option.

asy

References

[BHW96] F.C. Berger, A.H.M. ter Hofstede, and Th.P. van der Weide. Supporting Query by Navigation. In R. Leon, editor, *Information retrieval: New systems and current research, Proceedings of the 16th Research Colloquium of the British Computer Society Information Retrieval Specialists Group*, pages 26–46, Drymen, Scotland, EU, 1996. Taylor Graham.

[BPW93] C.A.J. Burgers, H.A. Proper, and Th.P. van der Weide. Organising an Information System as Stratified Hypermedia. In H.A. Wijshoff, editor, *Proceedings of the Computing Science in the Netherlands Conference*, pages 109–120, Utrecht, The Netherlands, EU, November 1993.

[CH94] L.J. Campbell and T.A. Halpin. Abstraction Techniques for Conceptual Schemas. In R. Sacks-Davis, editor, *Proceedings of the 5th Australasian Database Conference*, volume 16, pages 374–388, Christchurch, New Zealand, January 1994. Global Publications Services.

[HP95] T.A. Halpin and H.A. Proper. Subtyping and Polymorphism in Object-Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HPW94] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In G. Gupta, editor, *Seventeenth Annual Computer Science Conference*, volume 16 of *Australian Computer Science Communications*, pages 157–167, Christchurch, New Zealand, January 1994. University of Canterbury. ISBN 047302313

[HPW96] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Query formulation as an information retrieval problem. *The Computer Journal*, 39(4):255–274, September 1996.

[Pro94a] H.A. Proper. *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1994. ISBN 909006849X

[Pro94b] H.A. Proper. Interactive query formulation using point to point queries. Asymetrix Research Report 94-1, Asymetrix Research Laboratory, University of Queensland, Brisbane, Australia, 1994.

[Pro94c] H.A. Proper. Interactive query formulation using spider queries. Asymetrix Research Report 94-2, Asymetrix Research Laboratory, University of Queensland, Brisbane, Australia, 1994.

[Pro94d] H.A. Proper. Introduction to formal notations. Asymetrix Research Report 94-0, Asymetrix Research Laboratory, University of Queensland, Brisbane, Australia, 1994.

[PW95] H.A. Proper and Th.P. van der Weide. Information Disclosure in Evolving Information Systems: Taking a shot at a moving target. *Data & Knowledge Engineering*, 15:135–168, 1995.

[WMP^{+}76] A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.T. Meertens, and R.G. Fisker. *Revised Report on the Algorithmic Language ALGOL 68*. Springer-Verlag, Berlin, Germany, 1976.

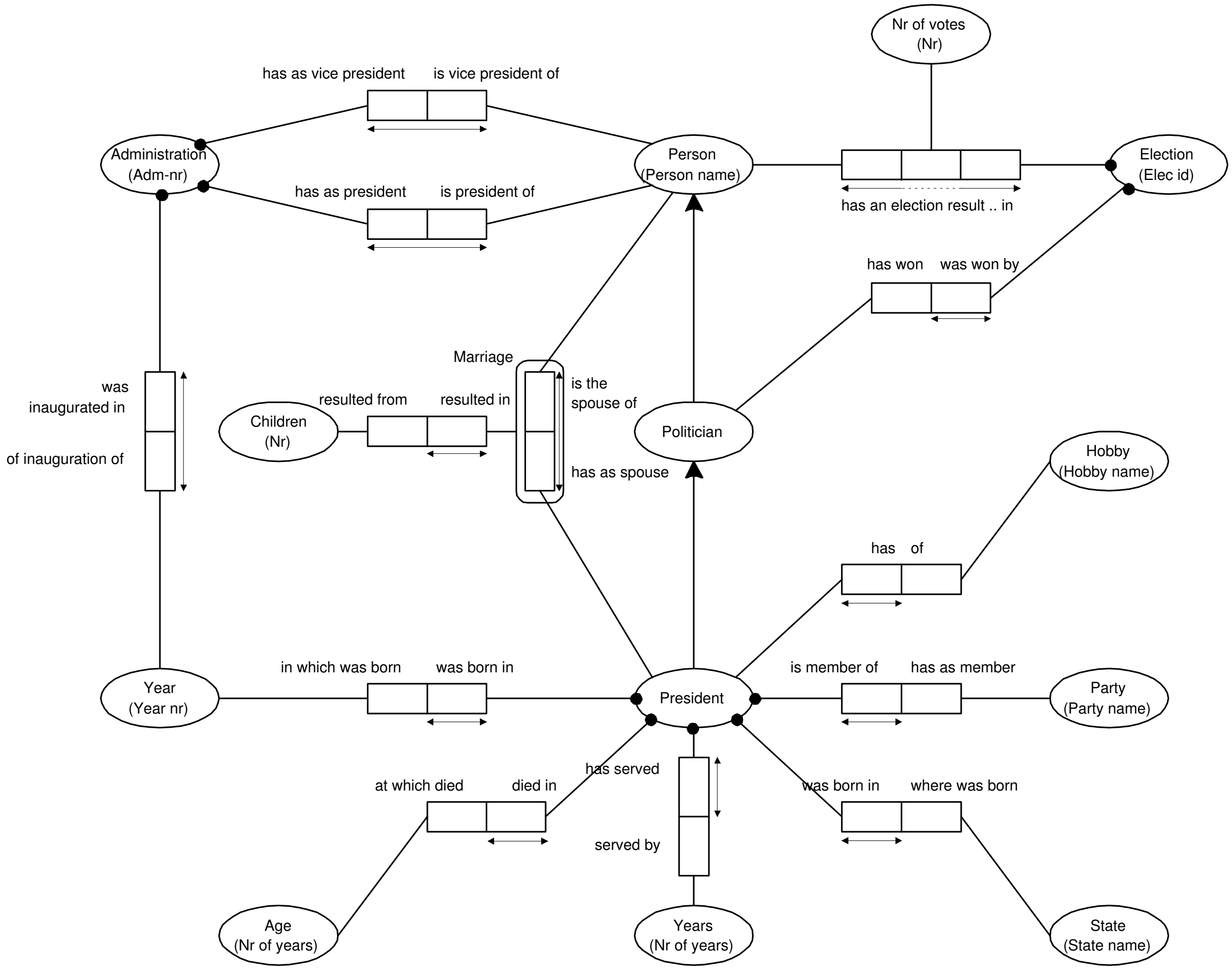


Figure 1: The structure of the presidential database

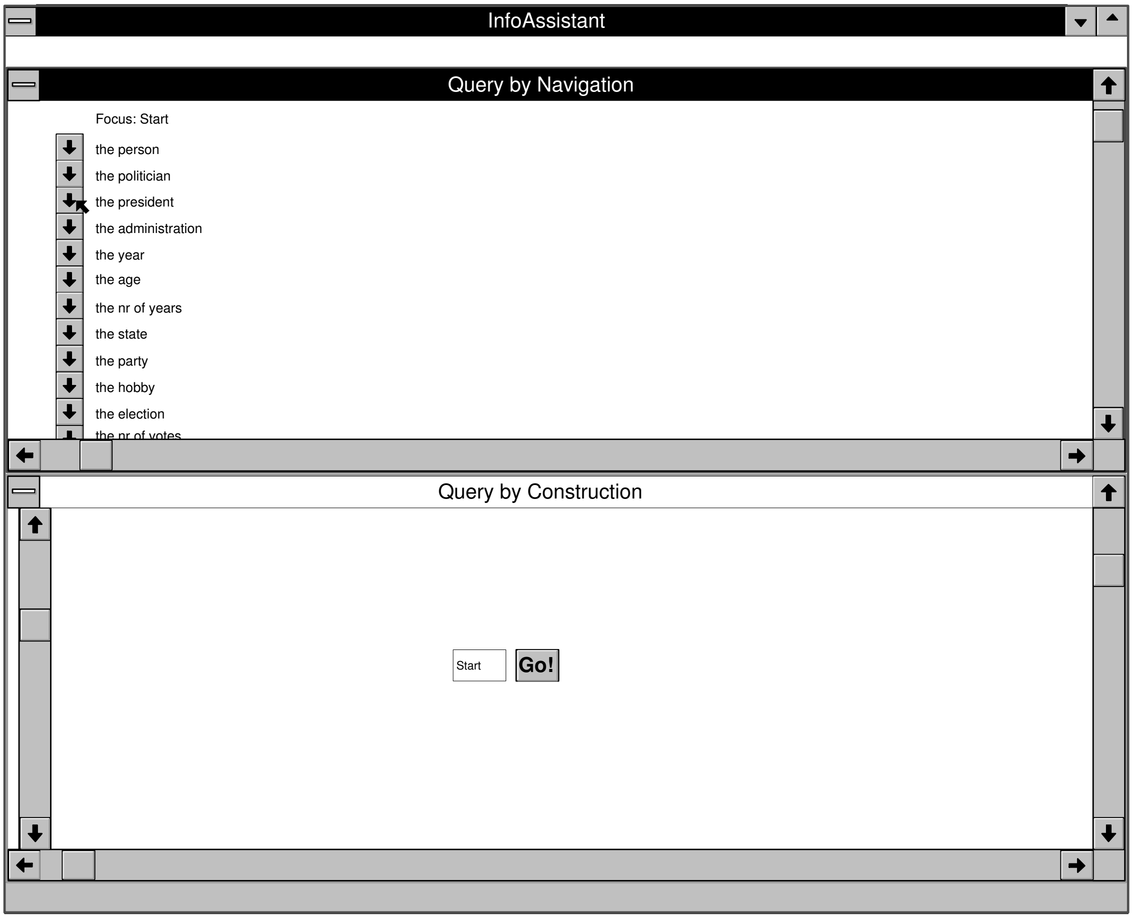
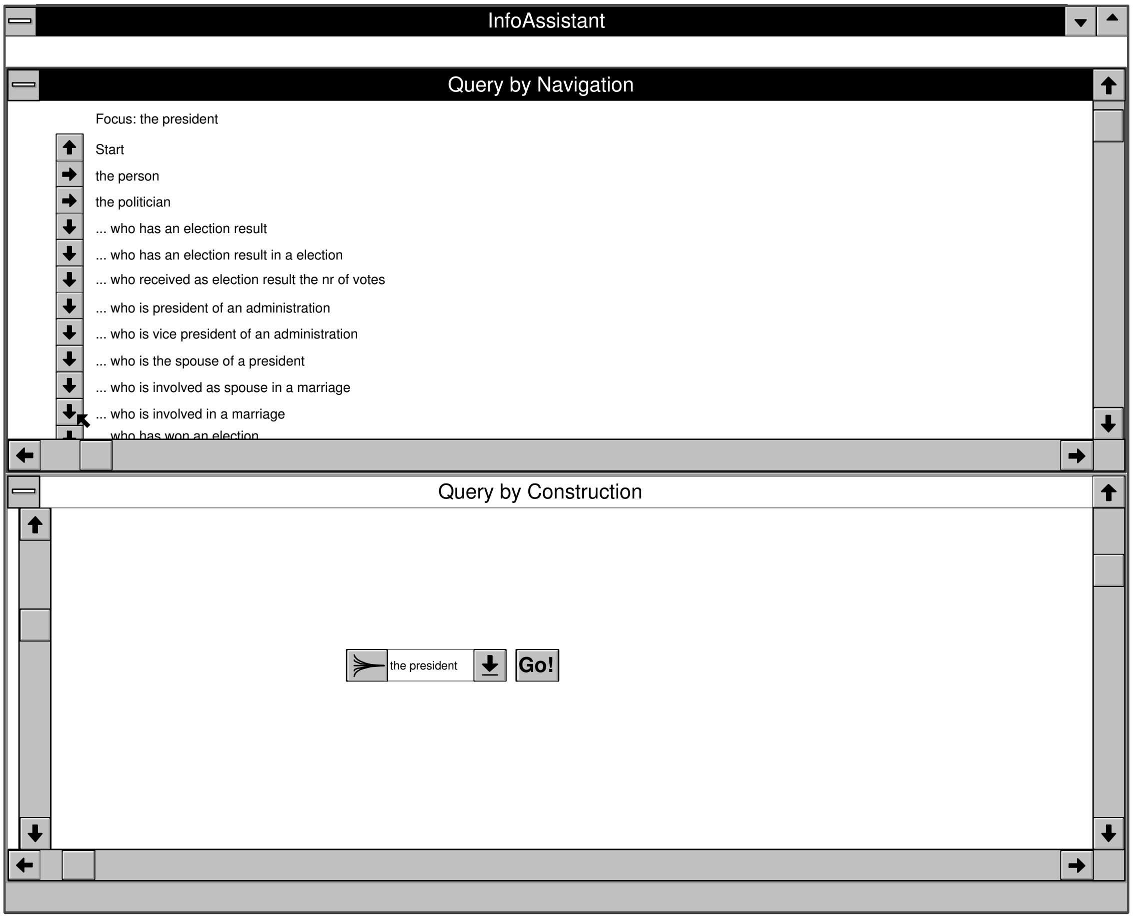


Figure 2: The starting node of the hyperindex



Focus: the president

Start

the person

the politician

... who has an election result

... who has an election result in a election

... who received as election result the nr of votes

... who is president of an administration

... who is vice president of an administration

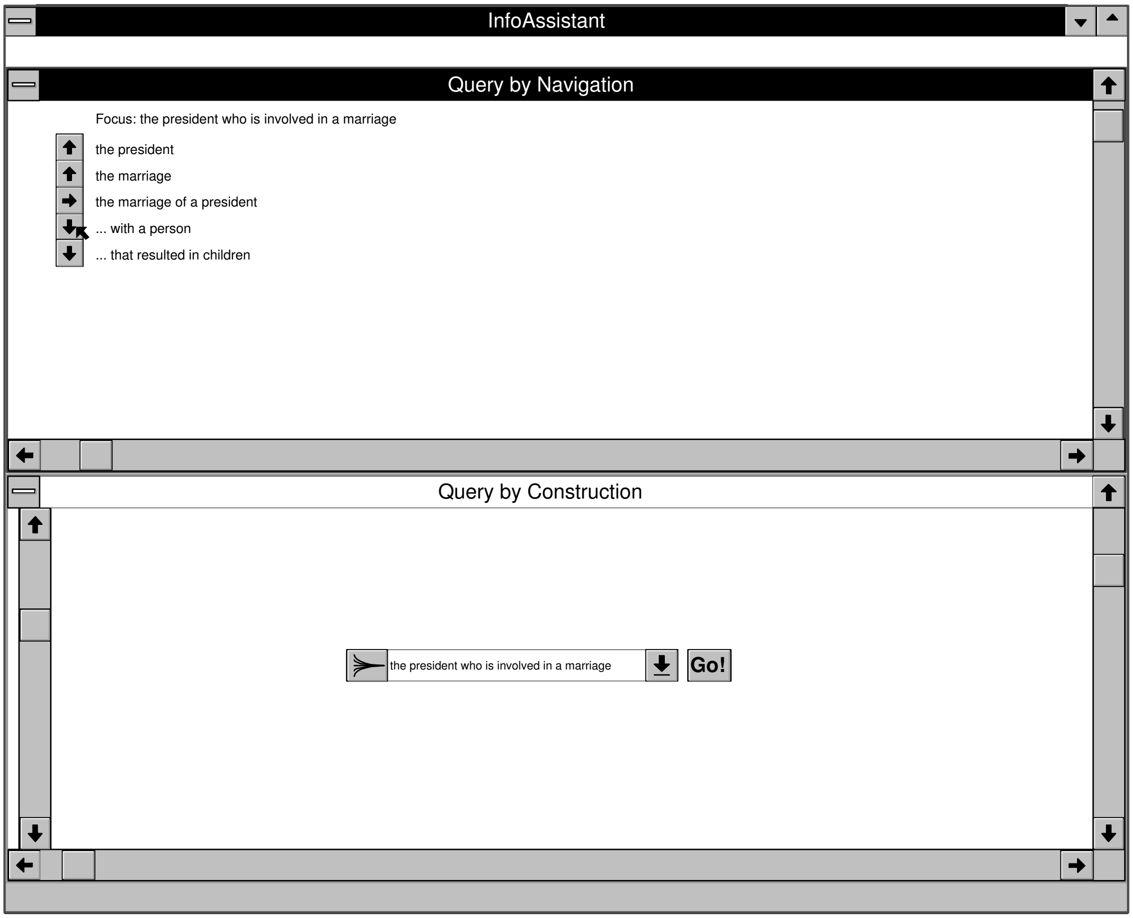
... who is the spouse of a president

... who is involved as spouse in a marriage

... who is involved in a marriage

who has won an election

Figure 3: The quest for a president who is married to a person

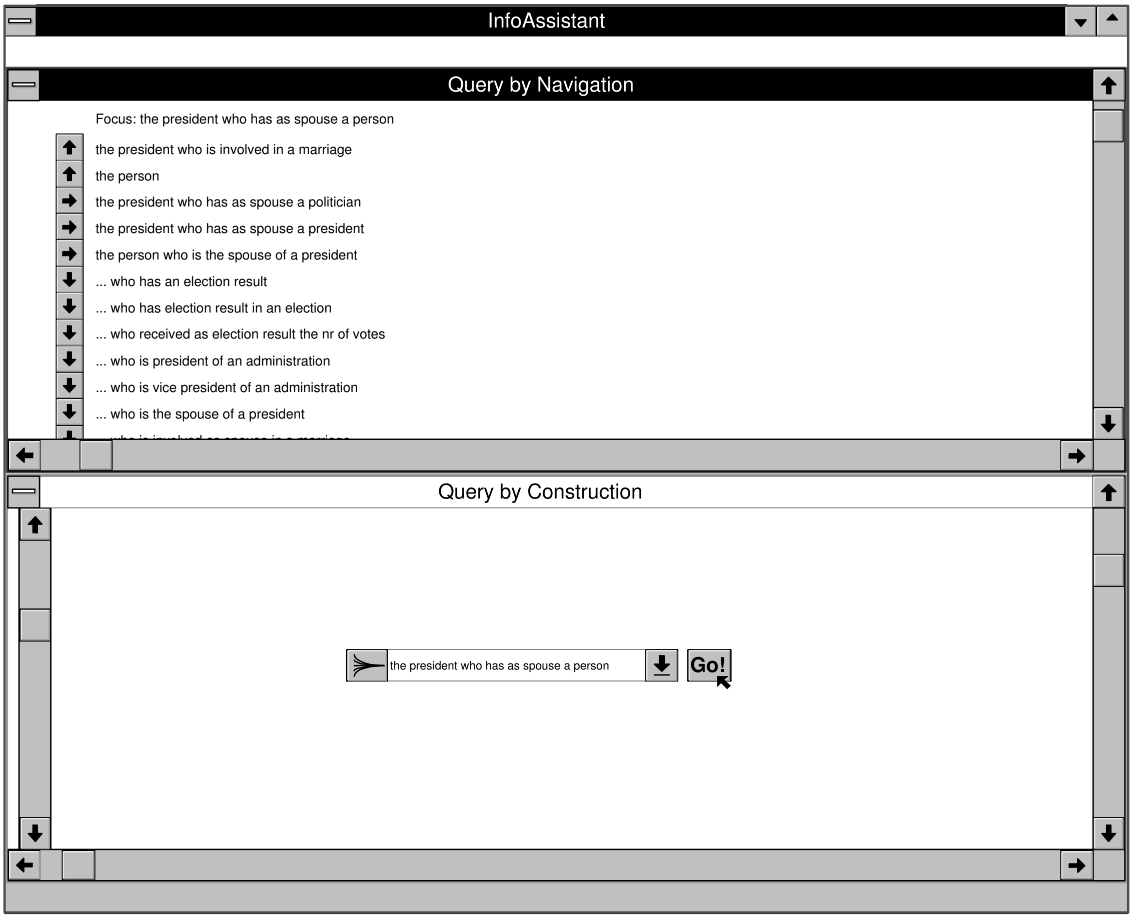


Focus: the president who is involved in a marriage

- the president
- the marriage
- the marriage of a president
- ... with a person
- ... that resulted in children

the president who is involved in a marriage

Figure 4: Focus on the objectified marriage relationship



Focus: the president who has as spouse a person

↑ the president who is involved in a marriage
↑ the person
→ the president who has as spouse a politician
→ the president who has as spouse a president
→ the person who is the spouse of a president
↓ ... who has an election result
↓ ... who has election result in an election
↓ ... who received as election result the nr of votes
↓ ... who is president of an administration
↓ ... who is vice president of an administration
↓ ... who is the spouse of a president
↓ ... who is involved as spouse in a marriage

the president who has as spouse a person

Figure 5: Preleminary result in the hyperindex