The following full text is a publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/28318

Please be advised that this information was generated on 2019-04-11 and may be subject to change.
Query Formulation as an Information Retrieval Problem

A. H. M. ter Hofstede¹, H. A. Proper²,³ and Th.P. van der Weide⁴

¹Department of Computer Science, The University of Queensland, 4072,
²Knowledge Discovery Research Group, Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane, 4001,
³Research Discovery Unit, Research Data Network Cooperative Research Centre, Level 7, Gehrmann Laboratories, The University of Queensland, Brisbane, 4072, Australia and
⁴Computing Science Institute, University of Nijmegen, Toernooiveld, 6525 ED Nijmegen, The Netherlands

Email: arthur@cs.uq.edu.au

Query formulation in the context of large conceptual schemata is known to be a hard problem. When formulating ad hoc queries users may become overwhelmed by the vast amount of information that is stored in the information system; leading to a feeling of lost in conceptual space. In this article we develop a strategy to cope with this problem. This strategy is based on ideas from the information retrieval world, in particular the query by navigation mechanism and the stratified hypermedia architecture. The stratified hypermedia architecture is used to describe the information contained in the information system on multiple levels of abstraction. When using our approach to the formulation 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 query formulation. Once users have specified a number of these linear paths, they may combine them to form more complex queries. This last process is referred to as query by construction and corresponds to the constructive phase of the query formulation process.

Received February 8, 1995; revised April 26, 1996

1. INTRODUCTION

Most present day organizations make use of some automated information system. This usually means that a large body of vital corporate information is stored in these information systems. As a result an obvious, yet crucial, function of information systems is the support of disclosure of this information. Without a set of adequate information disclosure avenues an information system becomes worthless since there is no use in storing information that will never be retrieved.

Adequate support for information disclosure, however, is far from a trivial problem. Most information systems do not provide any support for the users in their quest for information. Furthermore, the conceptual schemata of real-life applications tend to be quite large and complicated. As a result, the users may easily become 'lost in conceptual space' and they can end up retrieving irrelevant (or even wrong) objects and may miss out on relevant objects. Retrieving irrelevant objects leads to a low precision, missing relevant objects has a negative impact on the recall [1].

The disclosure of information stored in an information system has some clear parallels to the disclosure problems encountered in document retrieval systems. To draw this parallel in more detail, we quote the information retrieval paradigm as introduced in [1, 2]. The paradigm starts with an individual or company having an information need they wish to fulfill. This need is typically a vague notion and needs to be made more concrete in terms of an information request (the query) in some (formal) language. Formulation of this need leads to an information request q. The information request should be as good as possible a description of the information need. The information request is then passed on to an automated system, or a human intermediary, which will then try to fulfill the information request using the information stored in the system. To this end, the information request is matched against the characterization of information objects which are available in the information base (also referred to as information carriers or documents). This is illustrated in the information disclosure or information retrieval paradigm, presented in Figure 1 which is taken from [3].

The information retrieval paradigm for document retrieval systems is, in our opinion, directly applicable to traditional information systems. In the paradigm, the retrievable information is modeled as a set $K$ of information objects, which together constitute the information base (or population). In a document retrieval system the information base will be a set of documents [1, 2], while in the case of an information system the information base will contain a set of facts conforming to a conceptual schema. Each information object $o \in K$ is characterized by a set of descriptors $\chi(o)$ that facilitates its disclosure. The characterization of
Information objects is carried out by a process referred to
as indexing. In an information system, the stored objects
(the population or information base) can always be
identified by a set of (denotable) values; the identification
of those objects. For example, an address may be
identified as a city name, street name and house number.
The characterization of objects in an information system
is directly provided by the reference schemes of the object
types.

The actual information disclosure is driven by a
process of matching. In document retrieval applications
this matching process tends to be rather complex.
Furthermore, the characterization of documents is
known to be a hard problem [1, 4, 5] although newly
developed approaches turn out to be quite successful [6].
In information systems the matching process is less
complex as the objects in the information base have a
more clear characterization (the identification). In this
case, the identification of the objects (facts) is simply
related to the query formulation \( q \) by some (formal)
query language.

The remaining problem is then the query formulation
process itself. An easy and intuitive way to formulate
queries is absolutely essential for adequate information
disclosure. Quite often, the quest from users to fulfil their
information need can be aptly described by [7]:

\[ I \text{ don't know what I'm looking for, but I'll know when I find it. } \]

In document retrieval systems this problem is attacked
by using query by navigation [3, 8–11] and relevance
feedback mechanisms [12]. The query by navigation
interaction mechanism between a searcher and the
system is well-known from the information retrieval
field, and has proven to be useful. The above discussed
parallel between information disclosure in the context of
information systems, as well as information retrieval
systems, leads to the natural conclusion that these
mechanisms also apply to the query formulation
problem for information systems.

In line with the above discussed information retrieval
paradigm and the notion of relevance feedback, a query
formulation process (both for a document retrieval
system and an information system) can be said roughly
to consist of the following four phases.

1. The explorative phase. What information is there, how
   is it related and what does it mean?
2. The constructive phase. Using the results of phase 1,
   the actual query is formulated.
3. The feedback phase. The result from the query
   formulated in phase 2 may not be completely
   satisfactory. In this case, phases 1 and 2 need to be
   re-done and the result refined.
4. The presentation phase. In most cases, the result of a
   query needs to be incorporated into a report or some
   other document. This means that the results must be
   grouped or aggregated in some form.

Depending on the user's knowledge of the system, the
importance of the respective phases may change. For
instance, a user who has a good working knowledge of
the structure of the stored information may not require
an elaborate first phase and would like to proceed with
the second phase as soon as possible.

One important step that has already led to an
improvement of information disclosure in information
systems, has been the introduction of query languages on
a conceptual level. Examples of such conceptual query
languages are RIDL [13], LISA-D [14, 15] and FORML
[16]. Thus far, these languages are mainly used in the
context of the ORM [16–18] approach to information
modelling. A key difference between ORM approaches
and ER-based approaches are the close ties to natural
language. In the design procedures for ORM models, one
starts out by modelling a domain in terms of a set of
natural language sentences. These natural language
sentences verbalize this domain in terms used by the domain experts, i.e. the people who will be using the information system. Essentially, these verbalizations provide the basic expressions for the conceptual query languages. Similar attempts for ER models are usually hampered by the lack of rich verbalizations in such models.

By letting users formulate queries on a conceptual level, users are safeguarded from having to know the exact mapping to underlying relational tables to be able to formulate queries in a non-conceptual language like SQL. This allows users to concentrate on the actual query formulation, during the constructive phase, rather than on such mapping details. The next step on this path is to introduce ways to support users in the formulation of queries in such conceptual query languages (CQL); better exploiting the rich semantics provided by the conceptual schema of an application.

In this paper, we discuss how the application of query by navigation may lead to an elegant query formulation process. The status of the work is mainly theoretical. We realize that further empirical tests are needed to validate the use of query by navigation in this context. The use of query by navigation as proposed in this article, is expected to be particularly useful for the explorative and feedback phases of the query formulation process. We also introduce a query by construction mechanism allowing for the formulation of the final query using computer support.

The structure of the paper is as follows. In Section 2 we describe a general architecture for hypermedia systems used for query by navigation systems, and discuss how a conceptual schema and its population fit in this architecture. A larger example will provide a preview of the capabilities of a query by navigation mechanism. In Sections 3 and 4 the formal definition of the query by navigation system in relation to a conceptual schema is given. The migration between abstraction levels and the underlying prerequisites are discussed in Section 5. Finally, Section 6 discusses the query by construction mechanism complementing the query by navigation mechanism with syntactically richer expressions.

2. AN ARCHITECTURE FOR INFORMATION SYSTEM EXPLORATION

Stratified hypermedia is an architecture in which information is organized via several layers of abstractions, allowing access to the information via each of these layers. Base layers contain the actual information, while the other layers provide descriptions (abstractions) of this information with the purpose to simplify access to those base layers, and to provide insight in specific characteristics of this information.

In this section we briefly explain this architecture and discuss an example of how this architecture can be applied in the context of query formulation for information systems. The remainder of this article then develops this idea in more exact and formal terms.

2.1. Stratified hypermedia architecture

Stratified hypermedia architecture, in its simplest form, is a two level hypermedia architecture as introduced in [3, 8, 9, 11, 19, 20]. The two-level architecture usually features a descriptive layer (the hyperindex) indexing the lower layer (the hyperbase). The hyperbase contains the actual information, whereas the hyperindex only provides an outline (characterization) of the stored information. A stratified hypermedia architecture supporting multiple layers of abstraction is discussed in for example [21].

Users of a hypertext application based on the stratified hypermedia architecture may be compared with explorers. When navigating within one layer of abstraction, users are completely free to follow any link between pieces of information, thus allowing them to connect relevant information in their own (subjective) way. By navigating within and between the layers, users slowly but surely increase their knowledge about relevant parts of the information stored in the system at their own accord and preferred level of abstraction. In doing so, at a proper level of abstraction, they become cognitively better equipped to descend to the lower abstraction levels and select the desired pieces of information. The stratified hypermedia architecture, and its accompanying query by navigation system, have proven to be useful in practical situations (see [22, 23]). This creates the expectation that a similar query formulation strategy would also work well in the context of traditional information systems. Although there is currently no implementation available of the ideas presented here, there is considerable commercial interest. For example, the producers of the InfoModeler CASE-Tool [24] have developed a query formulation tool (InfoAssistant) which already incorporates some of the underlying ideas.

As an example of a hyperindex, consider Figure 2. This example is taken from [23] which describes a prototype implementation of a query by navigation based retrieval system that is still being used by history-of-art libraries, and is now sold as a commercial application. This is a simple example hyperindex, only dealing with a breakdown of the index expression

proclamation of resurrection of Jesus by disciples

In reality, a hyperindex is formed by the union of a large number of such smaller lattices, which then form a so-called lithoid. A sample navigation session is provided in Figure 3. A user starts at the starting node, which contains a list of all elementary terms from the hyperindex. The user can then select one of these words as a first refinement. Once a more complicated index expression has been selected, e.g. resurrection of Jesus, it becomes possible to select the more elementary expressions that are part of the currently focused
expression. In the case of resurrection of Jesus this would be resurrection and Jesus. In such a navigation session, the user basically traverses edges in the graph of the hyperindex as shown in Figure 2.

Each entry in the nodes displayed in Figure 3 represents one way to continue the search through the hyperindex. 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. An example of such a factor is the user's past search behaviour [25]. In this article we also briefly discuss another factor, which is based on the conceptual relevance of object types and other components within a conceptual schema.

In stratified hypermedia architecture, layers are related to each other via so-called characterizations. For instance, chunks of information from the hyperbase may be characterized by keywords. In that case, similarly to the given example, the hyperindex is organized around the concept of keyword. This characterization relation forms the basis for interlayer navigation. Typically, a user formulates an information request by navigating through the hyperindex to an information descriptor aptly describing the user's information need. Then the user transfers to the hyperbase, focusing on the information objects matching this description. In terms of the example given above, the user navigates in a query by navigation session from the start node to the node resurrection of Jesus. A transfer to the hyperbase would now lead to the presentation of a set of documents regarding the resurrection of Jesus; or at least the documents that have a characterization that matches resurrection of Jesus. The reverse interlayer navigation (hyperbase to hyperindex) transfers the user to a node listing all characterizations for the current object in the hyperbase. Each listed characterization provides a starting point for a further navigation session in the hyperindex.

Transferring to the hyperindex after arriving at an interesting node in the hyperbase corresponds to a search strategy where the user first searches an information object which is felt to be a typical example of the information need, and then asks for all similar information objects (query by example). This latter process involves a transfer from the object in the hyperbase that represents the information need, to an object in the hyperindex that characterizes this hyperbase object. This transfer from hyperbase to hyperindex is immediately followed by a transfer back to the hyperbase resulting in all objects in the hyperbase which are relevant to the current characterization of the hyperindex object.

Switching between layers can also be used as a feedback mechanism as used in information retrieval [26, 27]. When, after navigating through the hyperindex, the user finds that the current focus is possibly a proper description of the information need, they can test this by asking the system to present (part of) the relevant objects in the hyperbase. This way the user will, rather than completely transferring from the hyperindex to the hyperbase, get a first impression of what has been achieved so far. If the user is not satisfied, further refinements can be made based on the feedback provided by the system.

2.2. A formalization of stratified hypermedia architecture

This brings us to the formal introduction of stratified
hypermedia architecture as a network of layers that are related by characterization relations. A layer is introduced as a structure $(F, N, G, V)$ where:

1. $F$ is the set of (information) fragments, called the fragment base. Fragments are elementary parts of the stored information which can (will) not be decomposed structurally into smaller components.

   In the example given above, $F$ would at least contain:

   Jesus, resurrection, ..., proclamation of resurrection, ...

   proclamation of resurrection of Jesus by disciples

2. $N$ is the set of presentations (or nodes), referred to as the node base. Nodes are units of presentation and are used to present structural elements to the user. They are constructed from information fragment references, and appear in some ordered fashion within this presentation. Formally, a presentation is a structure $(F, \rho)$, where $F$ is a set of information fragments, and $\rho$ a partial order over $F$.

   In Figure 3 three example nodes are shown.

3. $G$ is a structure $(E, P)$, where $E$ is a set of syntactic categories, and $P$ a set of production rules. $G$ is referred to as the schema of the layer. The grammar $G$ is used to structure the information in a layer. Usually, this grammar is provided as a context-free grammar (SGML, ODA, HTML).

   In our example, the grammar we use is the grammar for the expressions associated to each of the nodes of the hyperindex from Figure 2. These expressions are essentially noun-phrases and are also referred to as index expressions [5, 28, 29].

4. $V$ is a set of views, called the mask. This is explained below.

Documents may be considered from different points of view, where each view features its own structure of the document. For example, a document may be considered as a (long) piece of text, or may be viewed as composed of sections, subsections, etc. It is the user who decides what view best suits the intended use of the document. As a result, within a view, each abstract information element belongs to some syntactic category. A set of composition rules describes the composition of syntactic categories in terms of other syntactic categories. Elementary syntactic
categories have associated an information fragment, i.e.
an elementary chunk of information. A view \( V \), in its
turn, is a structure \((S, M, \omega, \pi, L)\) where:

1. \( S \in E \) is the overall syntactic category for documents
   within this view (the start symbol of the subgrammar).

In the example case, this is the start symbol of the
index expression (noun-phrase) grammar.

2. \( M \) is the set of instantiations of syntactic categories
   that are available within this view. Such an instantia-
tion is called a molecule.

The nodes in the hyperindex graph shown in Figure 2
are examples of molecules; not to be confused with
the presentation nodes as depicted in Figure 3.

3. \( \omega \) is a binary relation over \( M \) that describes how
   syntactic categories are composed from each other,
   respecting the rules from \( G \). \( \omega \) is also referred to as the
   actual structure of the view.

In the example, \( \omega \) corresponds to the set of edges in
Figure 2.

4. \( \pi : M \rightarrow N \) maps each molecule from \( M \) to a pres-
etation unit.

The node from the hyperindex graph in Figure 2
labelled resurrection is presented by the node with the
same label. In the presentation node, the direct
environment of the resurrection node is displayed as
well, and the user is offered the possibilities to travel
downwards or upwards in the hyperindex.

5. \( L \) is a binary relation over \( M \) that defines a set of
   associative links. Links are used to describe cross-
   reference relations between documents. A particular
   link scheme consists of a set of links of the same
category. For example, an isa-relation (link schema)
might express the categorial classification of index
terms, while a co-relation is symmetric relation,
expressing that an index term corresponds to another
index term.

In the example, the presentation node labelled
resurrection of Jesus contains an associative link to a
node called resurrection of Christ.

Layers are related via characterization relations. A
characterization relates layers via specific views within
these layers and describes how molecules of one view are
associated with molecules of the other view. So, if \( \chi \) is a
characterization of layer \( L_1 \) via view \( V_1 \) into view \( V_2 \) of
layer \( L_2 \), then:

\[ \chi \subseteq V_1.M \times V_2.M \]

Note that more characterization relations may exist
between two layers, for example, originating from
different views within these layers.

Two kinds of navigations between molecules are
presented in Figure 4, two parse trees of the hyperbase
layer are presented \((B_1, B_2)\) and two parse trees of the
hyperindex layer \((I_1, I_2)\). The movement from one
molecule to another, using the underlying structure of
a parse tree, is called structural navigation. Selecting an
associative link initiates traversal of such a link, called

\( \text{FIGURE 4. Operations of the hypermedia.} \)

associative navigation, and leads to a change in context
(parse tree). Associative links are used to feature
cross-references between a fragment in one molecule
and a fragment in another molecule. The beam up
and beam down operations are used to facilitate
interlayer navigation.

2.3. Exploring an information structure

Now we have discussed the stratified hypermedia
architecture both formally and informally, we can
more concretely discuss the ideas presented in this
article. To this end an example query formulation
process is discussed that utilizes the stratified hypermedia
architecture. The remainder of this article is concerned
with a proper formalization of these ideas.

In our view, the process of query formulation
 corresponds to a search through the information
system gradually to fulfil some information need.
Using query by navigation, the arguments are con-
structed that are to be integrated by the constructive
phase of query formulation. During query by navigation,
a (partial) query is formulated by stepwise refining or
enlarging the current formulation (the focus), until the
searcher is satisfied with the current formulation. In the
example we make use of the conceptual schema of a
database for presidents of the United States of America
as depicted in Figure 5. This schema is provided as an
Object-Role Modelling schema \([16,30]\). It describes a
domain in terms of relationship types (also referred to as
fact types) and object types. Roles (the rectangular
boxes) indicate how object types participate in relation-
ship types. Some examples of relationship types are: ...
has vice president ..., ... has of ... in ..., and examples of
object types are: Administration, Person. On some types a
subtyping relationship is present. President is subtype of
Politician, which in its turn is a subtype of Person. Besides

Let us presume the searcher selects the president who is involved in a marriage as the next focus. This action leads to the node depicted in Figure 8. This node shows a second class of associative links. Besides associative links resulting from subtyping, we also distinguish associative links to the reversed formulation of the current focus, i.e. the marriage of a president. When navigating through the hyperindex, refinements to the current focus will take place on the tail of the current focus. By reversing the current focus, refinements can be made on the front as well.

The user decides to select the refinement with as spouse the person, leading to Figure 9. The user considers this, for the moment, to be a proper description of the information need. To get an impression of the query result so far, the user selects the beam down option. This results in the node depicted in Figure 10. This node is neither part of the hyperbase, nor is it part of the hyperindex. It is an ad hoc node representing the result of the focus of Figure 9 interpreted as a query on the underlying database. The user can now select an instance for further navigation, which will then indeed take place.
FIGURE 6. The starting node of the hyperindex.

in the hyperbase. Let us presume the user selects the marriage between president Washington and M. D. Custis as starting point for further refinement.

This leads to the node shown in Figure 11. This node shows as refinements all information known about M. D. Custis. Since the database only focuses on presidents, there is no other information known about M. D. Custis other than that she is married to president Washington. The only associative link for this node leads to the objectification of the marriage instance. In the original ORM schema of Figure 5, the object type Marriage is modelled as a so-called compositely identified object type. This follows from the constraint pattern and in particular from the inter-relationship uniqueness constraint, depicted as an encircled U. It means that a marriage is identified as a combination of a president and a person. This allows us to treat a combination of a president (Washington) and a person (Custis) as an instance of the object type Marriage.

The user now realizes that s/he wanted to know more about Washington’s marriage, remembering the initial interest in the number of children of presidential marriages. Therefore, the user now selects the objectification of the marriage as the next point in the navigation. The resulting node, shown in Figure 12, shows all information known about this marriage. Since the user was interested in the number of children born in presidential marriages, the user selects the resulted in children refinement. This leads to the node depicted in Figure 13. The user is now satisfied with the description of the information need in that the instances (marriage and number of children) are good examples of the desired kind of information.

The user could now select the beam up operation to end up at the more general description of the current focus: the marriage that resulted in children. The user can then continue with a beam down to end up with all marriages and the resulting number of children. This latter process of:

1. a beam up of an information object in the hyperbase to a more general description in the hyperindex,
2. followed by a beam down back to the hyperbase that results in all objects with a similar characterization as the original information object

is an example of query by example. Query by example [31] allows users to specify an example of the approximate query they wish to see answered. The system will then try to induce other similar results from this example.

3. THE HYPERINDEX LAYER

To formally define the stratified hypermedia that can be associated to a conceptual schema of an object-role modelling technique, we first need some formal definition of what an ORM conceptual schema is. The formalization used here is a simplified version of the formalization of ORM as provided in [14].
3.1. Generalized ORM schemas

A conceptual schema is built around a set of object types \( \mathcal{O} \) and fact types (also called relationship types): \( \mathcal{F} \). Each fact type \( f \) consists of a set of roles from \( \mathcal{P} \). The fact types in \( \mathcal{F} \) should form a partition of the roles in \( \mathcal{P} \). Due to this partition, the function \( \text{Fact}: \mathcal{P} \rightarrow \mathcal{F} \) for roles can be defined for any \( r \in \mathcal{P}, f \in \mathcal{F} \):

\[
\text{Fact}(r) = f \iff r \in f
\]

All roles have a base, the object type playing the role. This base is provided by the function \( \text{Base}: \mathcal{P} \rightarrow \mathcal{O} \).

All object-role modelling variations allow for the definition of type hierarchies. Different ways to introduce such hierarchies exist, for example specialization and generalization (polymorphism) [14, 32–34]. For our purposes we can simply presume the existence of a general notion of a type hierarchy which could involve different flavours of inheritance. The relation \( \text{IdfBy} \subseteq \mathcal{O} \times \mathcal{O} \) is used to capture this general notion. The intuition is that if \( x \text{IdfBy} y \), object type \( x \) is identified through object type \( y \), which means that \( x \) inherits properties from \( y \). It should be clear that this general relation can be used to capture both
The marriage of 'Washington G.' and 'Custis M.D.'

- president 'Washington G.'
- person 'Custis M.D.'
- ... resulted in 0 children

FIGURE 12. Information about the marriage between Washington and Custis.

The marriage of 'Washington G.' and 'Custis M.D.' resulted in 0 children

- the marriage of 'Washington G.' and 'Custis M.D.'
- 0 children
- ... resulted from the marriage of
- 'Madison J.' and 'Todd D.D.P.'
- 'Jackson A.' and 'Robards R.D.'
- 'Polk J.K.' and 'Childress S.'

FIGURE 13. Resulting node.

generalization (polymorph) and specialization relations. The \( \text{IdfBy} \) relation is presumed to be transitive. Furthermore, we use \( x \text{IdfBy} y \) as an abbreviation for \( x \text{IdfBy} y \lor x = y \) and \( x \sim y \) as an abbreviation for \( x \text{IdfBy} y \lor y \). The \( \sim \) relationship is the so-called type relatedness relationship. Two types are type related if their populations may share instances.

As an example, consider the ORM schema depicted in Figure 14. In this schema we have:

\[
\begin{align*}
\mathcal{O} &= \{A, B, C, D\} & \mathcal{F} &= \{f, g\} \\
\mathcal{P} &= \{p, q, r, s\} & \text{IdfBy} B \\
f &= \{p, q\} & g &= \{r, s\} \\
\text{Base}(p) &= A & \text{Base}(q) &= B \\
\text{Base}(r) &= C & \text{Base}(s) &= D
\end{align*}
\]

A further simplification made to the formalization of ORM models, besides the unified treatment of specialization and polymorphism, is the treatment of objectification. In ORM modelling (and ER for that matter), one can choose explicitly to objectify fact types. For example, the schema fragment depicted in the left hand side of Figure 15 is equivalent to the fragment depicted in the right hand side. For instance, in the presidential database example the object type Marriage could have been modelled to conform the left hand side. In the query by navigation mechanism for ORM schemas presented in this paper, objectifications are treated as if they are flattened, i.e. as shown in the right hand side of Figure 15.

Using this brief formal description of ORM schemas, the stratified hypermedia architecture for ORM models in general (without limiting ourselves to one dialect only) can be built. It should be noted that the formal description equally applies to ER models, so the results of this paper can be translated to ER models as well. In [35] the close relationship between ER models and ORM models is discussed in more detail.

3.2. Linear path expressions

The backbone of the nodes in the hyperindex and hyperbase is formed by the so-called linear path expressions [14]. These expressions are built from object types, roles, and instances. All these components can be interpreted as binary relations and as such concatenated to each other. An object type \( o \) occurring in a path expression corresponds to a binary relationship with tuples \( (x, x) \) for every instance \( x \) of type \( o \). A role \( r \) corresponds to a binary relationship connecting \( \text{Base}(r) \) to \( \text{Fact}(r) \), with tuples \( (x, y) \) where \( x \) is the \( r \) part of fact instance \( y \). To traverse fact types in a path expression, it must be possible to reverse the order of the \( \text{Base}(r) \) and \( \text{Fact}(r) \) parts of a role. Therefore, \( r^- \) represents the reversed binary relation associated to role \( r \).

When displaying linear path expressions in the nodes of the query by navigation mechanism, the linear path expressions need to be verbalized. These verbalizations can be derived from the names given to the object types and roles from the conceptual schema. In this article we simply presume the existence of a function \( p \) verbalizing these linear path expressions. For a more detailed discussion on the verbalization of linear path expressions refer to [36,37].

3.3. The descriptive view grammar

Formally, a hyperindex is introduced as a structure \( \mathcal{L} = (\mathcal{F}, \mathcal{N}, \mathcal{G}, \mathcal{V}) \) (see also [36,38]). The fragment
base ($F_1$) of the hyperindex simply contains the elements for verbalizations of the linear path expressions. Therefore we should have:

$$F_1 \subseteq \text{Name}$$

Some examples from the fragment base of the presidential schema example are:

President, Marriage, involved in, Votes, is vice president of

The following step in defining the hyperindex is the introduction of its schema:

$$G_1 \triangleq \langle E_1, P_1 \rangle$$

In our approach, we navigate through the hyperindex by refinements and enlargements of a linear path expression corresponding to a molecule in the hyperindex. An alternative approach would be to define a grammar for the verbalization of these path expressions and navigate through the verbalizations rather than the underlying linear path expression.

The context-free production rules for the hyperindex ($P_1$) define the way in which linear path expressions can be extended. The grammar $G_p$ contains for each object type $x$ a corresponding nonterminal (syntactic category) ($P_x$). Instantiations of syntactic category ($P_x$) describe simple properties of (instances of) object type $x$, i.e. properties that can be derived via a linear path expression starting in object type $x$. For any $o \in O$ we have the following rules:

$$\langle S \rangle \rightarrow \langle P_o \rangle$$

$$\langle P_o \rangle \rightarrow x$$

The identification hierarchy leads to the following rules. If $x$ likesBy $y$, then:

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

which means that properties about $y$ may be used in expressions about $x$, but not vice versa. For roles $r$ and $q$ such that $r \neq q$ and Fact ($r$) = $r$, $q$ we have:

$$\langle P_{\text{Base}(q)} \rangle \rightarrow \langle P_{\text{Base}(q)} \rangle \circ r \circ \text{Fact}(r) \circ o q \circ o \text{Base}(q)$$

For the hyperindex we have the following syntactic categories: $E_1 \triangleq \{ \langle P_x \rangle | x \in O \} \cup \{ S \}$. Note that the above syntax describes meta-rules, which are concretized by substituting an actual object type for meta-nonterminal $x$ and roles $q$, $r$. So, basically this grammar is a two level grammar [39]. For the example ORM schema depicted in Figure 14 we have:

$$\langle S \rangle \rightarrow \langle P_A \rangle \circ \langle P_B \rangle \circ \langle P_C \rangle$$

$$\langle S \rangle \rightarrow \langle P_B \rangle \circ \langle P_A \rangle \circ A \circ \langle P_B \rangle \rightarrow B$$

$$\langle P_C \rangle \rightarrow C \circ \langle P_B \rangle \circ D \circ \langle P_C \rangle \rightarrow \langle P_B \rangle$$

$$\langle P_A \rangle \rightarrow \langle P_B \rangle \circ q \circ o p \circ \text{Base}(A)$$

$$\langle P_B \rangle \rightarrow \langle P_A \rangle \circ p \circ o q \circ \text{Base}(B)$$

$$\langle P_C \rangle \rightarrow \langle P_D \rangle \circ o q \circ o r \circ \text{Base}(C)$$

$$\langle P_D \rangle \rightarrow \langle P_C \rangle \circ o q \circ o r \circ \text{Base}(D)$$

### 3.4. Molecules in the index view

In the current approach, the hyperindex for a conceptual schema will contain only a single view. A view is formally introduced as a structure:

$$V_1 \triangleq \langle S_{\omega_1}, M_1, \pi_1, L_1 \rangle$$
The starting point of this view is $S_i \in E_i$, which is $(S)$. The molecules $M_i$ are formed by the set of linear path expressions augmented with the empty path expression $\epsilon$. Some path expressions can be proven to be structurally empty, i.e. in every population they yield an empty result. As these path expressions are not meaningful in this context, they are omitted from $M_i$.

The actual structure $\omega$ is a subset of $M_1 \times M_1$. Let $x$ be an object type and $P$ be a fragment of a path expression, then this set is identified by the following kinds of structural links:

1. A link from the empty path expression $\epsilon$ to any molecule $x$.
2. A link from a molecule $Px$ to a molecule $Py$ if $x \text{opBy } y$ capturing the identification hierarchy.
3. A link from molecule $P$ to molecule $\text{Rev}(P)$ if $P \neq \text{Rev}(P)$ catering for the reversal of path expressions.

The reversal of a path expression by $\text{Rev}$ is recursively defined as:

$$\text{Rev}(P \circ P \circ f \circ q^{-} \circ x) \triangleq x \circ q \circ f \circ p^{-} \circ \text{Rev}(P)$$

$$\text{Rev}(x) \triangleq x$$

An example of such a reversal is:

$$\text{Rev}(x \circ p \circ f \circ q^{-} \circ y) = y \circ q \circ f \circ p^{-} \circ x$$

In Figure 17 we have added some associative links (dotted lines) to the example hyperindex from Figure 16.

### 3.5. Presentation of molecules

Molecules are presented by nodes specified by $\pi_1$. As stated before, a molecule will be presented by a molecule containing the direct environment of the molecule. The environment of a node is depicted in Figure 18. A node, presenting molecule $M_i$, 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, calling the attention to related alternatives.

The presentation of a molecule is formally identified as:

$$\pi_1(M) \triangleq (\rho(M), \rho(\Delta(M)), \rho(\triangledown(M)), \rho(\triangleright(M)))$$

where the direct environment of $M$ is captured by:

$$\Delta(M) \triangleq \{N \mid (N, M) \in \omega_1\}$$

$$\triangledown(M) \triangleq \{N \mid (M, N) \in \omega_1\}$$

$$\triangleright(M) \triangleq \{A \mid (M, A) \in L_1\}$$
We presume that the verbalization function $\rho$ is extended to sets of path expressions in the natural way. The general format of a node is displayed in Figure 19.

The order in which the elements of the above sets are actually presented in a node can be based on a variety of factors. It can for example be based on the previous search behaviour of a user [25]. In the context of conceptual schemas, it also makes sense to base this order on conceptual relevance. When looking at a conceptual schema one can distinguish object types that are conceptually more relevant than others. This relevance can be captured as some numerical value expressing the conceptual relevance of some schema component. For instance, the object type President has, intuitively, a higher relevance than Hobby for the modelled domain. In, for example, [40, 41] a procedure is described to determine the most relevant object types in a given conceptual schema. Entries in a node could now be ordered based on the conceptual relevance of the object types occurring in an entry. Which ordering factors should actually be taken into consideration can only be determined after extensive empirical testing. Currently no implementation is available to provide such a testing environment.

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 verbalization rules, refer to [36] or [37].

4. THE HYPERBASE LAYER

In this section we describe the organization of the hyperbase layer for a conventional information system. This layer is internally organized according to the associated conceptual schema, and is instantiated in accordance with the population of that schema. The stratified hypermedia architecture contains a single view on the population of the associated conceptual schema, the so-called Base View. This view describes the complete information base in the format of instantiated linear path expressions. The translation of instantiations into a

FIGURE 17. Example hyperindex with associative links.

FIGURE 18. The environment of a molecule.

FIGURE 19. The presentation of a molecule.
hyperbase is also carried out bottom-up, the fragment base is defined first, followed by the node base, the schema and the views, respectively. As the hyperbase layer is quite similar to the hyperindex layer, we will not provide any additional examples.

4.1. Fragment base

The fragment base ($F_0$) of the hyperindex is formed from the names used in the verbalizations of the path expressions as is the case for the hyperindex, extended with the denotations of the instances (see Figure 11).

4.2. The base view grammar

The grammar rules for the hyperbase ($G_0$) are formed in the same way as for the hyperindex except for one rule. For any $o \in \mathcal{O}$ and instance $i$ of $o$ we have:

$$\langle P_\mathcal{X} \rangle \rightarrow x \circ i$$

rather than:

$$\langle P_\mathcal{X} \rangle \rightarrow x$$

For the hyperbase we therefore have the same set of syntactic categories: $E_0 \triangleq \{\langle P_\mathcal{X} \rangle | x \in \mathcal{O} \} \cup \{\mathcal{S}\}$. Similarly to the hyperindex, the starting symbol $S_0$ is again $\langle \mathcal{S} \rangle$. The set of molecules $M_0$ of the hyperbase corresponds to the path expressions that can be built from the grammar rules in $G_0$ and which are not empty.

Let $x$ be an object type, let $i, j$ and $k$ be instances, and let $P$ be a fragment of a path expression, then the structure $\omega_0$ of the hyperbase is provided as:

1. A link from the empty path expression $e$ to any molecule $x \circ i$.
2. A link from a molecule $Px \circ i$ to a molecule $Px \circ o \circ j \circ o \circ q \circ o Base(q) \circ k$ if $r \neq g$, $Fact(r) = \{r, q\}$ and $x \text{ idfBy Base}(r)$.

Note by requiring that the molecules in $G_0$ correspond to path expressions which do not have an empty result, the instances $i, j$ and $k$ are implicitly required to be associated via the used relationship type $Fact(r)$.

The associative links ($L_0$) for the hyperbase are quite similar to the links in the hyperbase. Let $x, y$ be object types and $i$ an instance, then we have the following kinds of associative links:

1. A link from a molecule of the form $Px \circ i$ to a molecule $Px \circ o \circ i$ if $x \sim y$.
2. A link from molecule $P$ to molecule $Rev(P)$ if $P \neq Rev(P)$ catering for the reversal of path expressions.

4.3. Presentation of molecules

The presentation of molecules from the hyperbase does not differ from the presentation of molecules from the hyperindex. The verbalization function $\mathcal{V}$ should also be able to handle instances in linear path expressions.

5. RELATING HYPERBASE AND HYPERINDEX

In this section, the hyperbase and hyperindex are related by using the beam up and beam down operations. We do this in a top down fashion. First we discuss how hyperbase and hyperindex can be compared with each other presuming some characterization for the molecules in both layers is provided. This is followed by a discussion of three possible ways of characterizing molecules, leading to strong, hybrid and weak characterizations.

5.1. Interlayer navigation

In general, interlayer navigation is a way to operationalize the characterizations of molecules in the (differing) involved layers. In StratArch the characterization $\chi$ of layer $L_1$ via view $V_1$ into view $V_2$ of layer $L_2$ has been introduced as:

$$\chi \subseteq V_1.M \times V_2.M$$

In the remainder of this section this characterization function is defined in more detail.

For convenience, we assume that each view within any layer is identified by a unique number. The characterization of view $v$ in terms of view $w$ is denoted as $\chi_{v \leftrightarrow w}$. We number the standard view within the hyperbase layer as view number 0, and the standard view on the hyperindex layer as view number 1. Using characterizations we are able to relate molecules to each other to identify the relevance from one molecule to another. Several ways to determine the similarity between characterizations exist [2]. A simple and well-known method to compute similarity is the following formula, also well known as Jaccard's coefficient:

$$Sim(C_1, C_2) \triangleq \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

This formula is also useful in the context of multisets. In this case the similarity formula can also be written as:

$$Sim(C_1, C_2) \triangleq \sum \frac{\min(Freq(x, C_1), Freq(x, C_2))}{Freq(x, C_1) + Freq(x, C_2)}$$

where $Freq(x, M)$ returns the frequency of element $x$ in multiset $M$. Multisets are sometimes used to represent the result of queries (SQL, LISA-D). Another advantage is that multisets allow for a more adequate characterization, as they take frequencies into account.

Interlayer navigation may now be defined from the characterization relations by using the $Sim$ function. The beam operator $Beam_{n \rightarrow m} : M_n \rightarrow (M_m \rightarrow [0, 1])$ associates the molecules from view $n$ with the molecules from view $m$ in terms of the characterization function $\chi_{n \rightarrow m}$ in terms of relevance as follows:

$$Beam_{n \rightarrow m}(x) \triangleq \lambda y \in M_m.Sim(\chi_{n \rightarrow m}(x), \chi_{m \rightarrow n}(y))$$

By taking $n = m$ intra layer beaming results, and by taking $n \neq m$ inter layer beaming results. Beam from $a$
The strong characterization of a molecule (instantiated path expression) from the hyperbase in terms of molecules (path expressions) from the hyperindex consists of all queries that (i) are structurally compatible with the molecule to be characterized and (ii) from which this molecule may result in the query result. The characterization is effectively obtained by the replacement of all typed instances by all types which are associated to these instances.

Let \( x \) be an object type, \( i \) an instance, \( p \) a role and \( P, Q \) linear path expressions, then the strong characterization for the hyperbase is recursively defined as:

\[
\begin{align*}
\chi_{0 \rightarrow 1}^{h}(x \circ i) & \triangleq \{y \uparrow | x \sim y\} \\
\chi_{0 \rightarrow 1}^{h}(p) & \triangleq \{[p]\} \\
\chi_{0 \rightarrow 1}^{h}(p^{-}) & \triangleq \{[p^{-}]\} \\
\chi_{0 \rightarrow 1}^{h}(P \circ Q) & \triangleq \{[A \circ B \uparrow_{\text{nm}} | A \in \chi_{0 \rightarrow 1}^{h}(P) \land B \in \chi_{0 \rightarrow 1}^{h}(Q) \land \text{InPop}(P \circ Q, A \circ B)\}
\end{align*}
\]

where InPop is defined as:

\[
\text{InPop}(A, B) \triangleq \mu[A](\text{Pop}) \subseteq \mu[B](\text{Pop})
\]

This limits the \( A \circ B \) combinations to those that define a query that is a superset of the original query \( P \circ Q \). The expression \( P \circ Q \) contains instances whereas \( A \circ B \) contains none. This means that \( P \circ Q \) is (should be) more limiting than \( A \circ B \). Finally, expression \( x \uparrow^{m} \) refers to \( x \) with frequency \( n \), whereas \( x \in^{n} M \) is used to denote that \( x \) occurs in multiset \( M \) with frequency \( n \).

Please note again that since we limit ourselves to only two layers (hyperbase and hyperindex), the characterization is only provided for navigation between these two layers.

For the hyperindex, we have the characterization:

\[
\begin{align*}
\chi_{1 \rightarrow 0}^{h}(x \circ i) & \triangleq \{y \uparrow | x \sim y\} \\
\chi_{1 \rightarrow 0}^{h}(p) & \triangleq \{[p]\} \\
\chi_{1 \rightarrow 0}^{h}(p^{-}) & \triangleq \{[p^{-}]\} \\
\chi_{1 \rightarrow 0}^{h}(P \circ Q) & \triangleq \{[A \circ B \uparrow_{\text{nm}} | A \in \chi_{1 \rightarrow 0}^{h}(P) \land B \in \chi_{1 \rightarrow 0}^{h}(Q) \land \text{InPop}(P \circ Q, A \circ B)\}
\end{align*}
\]

As an example, the path expression \( B \circ i \circ o \circ f \circ o \circ j \circ o \circ k \) from the hyperbase of the schema in Figure 20 yields the following characterization: \( A \circ p \circ f \circ o \circ q \leftarrow \circ C \), \( B \circ o \circ f \circ o \circ q \leftarrow \circ C \). This set is also exactly the characterization of the path expression \( A \circ p \circ o \circ f \circ q \leftarrow \circ C \) from the hyperindex that can be associated to Figure 20.

A possible way to enhance this flavour of characterization, is to add a conceptual relevance value to each of the elements contained in a characterization. For a given expression in a characterization, the conceptual relevance can be defined as the average conceptual relevance of all types and roles occurring in this expression.

### 5.2.2. Hybrid characterization

The hybrid characterization of a molecule describes an instantiatied path expression in terms of the frequencies of all roles, reverse roles and object types in this expression. It is similar to the weighted vector model for conventional documents. A difference, however, is that type-related object types are also taken into account in the characterization. For the hyperbase we therefore have:

\[
\begin{align*}
\chi_{0 \rightarrow 1}^{h}(x \circ i) & \triangleq \{y \uparrow | x \sim y\} \\
\chi_{0 \rightarrow 1}^{h}(p) & \triangleq \{[p]\} \\
\chi_{0 \rightarrow 1}^{h}(p^{-}) & \triangleq \{[p^{-}]\} \\
\chi_{0 \rightarrow 1}^{h}(P \circ Q) & \triangleq \chi_{0 \rightarrow 1}^{h}(P) \cup \chi_{0 \rightarrow 1}^{h}(Q)
\end{align*}
\]

FIGURE 20. Example characterizations.
For the hyperindex this leads to:

\[
\chi\{| \mathbf{x} \in \mathbf{y} \} = \{ \{ y | y \sim \mathbf{x} \} \}
\]

\[
\chi_l\omega_1(p) = \{ \{ y \} \}
\]

\[
\chi_l\omega_1(p^c) = \{ \{ y^c \} \}
\]

\[
\chi_l\omega_1(p \circ \omega) = \chi_l\omega_1(p) \cup \chi_l\omega_1(q)
\]

For the example path expression from the hyperbase

\[
(A \circ p \circ o q \circ C)
\]

this leads to the following result: \{\{A, B, p, f, q^c, C\}\}. The hybrid characterization

of the (hyperindex) path expression: \( A \circ p \circ o q \circ C \) is also:

\[
\{\{A, B, p, f, q^c, C\}\}.
\]

This hybrid characterization flavour could also be enhanced by using conceptual relevance factors for the components.

5.2.3. Weak characterization

Both strong and hybrid characterization are based on the structure of instantiated path expressions. Instantiated path expressions are abstract objects that are concretized by their representation to the user. In the stratified

hypermedia architecture, this representation is captured by the presentation function \( \pi \). This provides the opportunity to make a characterization of instantiated path expressions based on their presentation. Presentations of instantiated path expressions are constructed from verbalizations of structural elements. These names are exploited and monitored (with their frequency) in the weak characterization:

\[
\chi^p_{\mathbf{M}}(\mathbf{M}) = \{ y^n | w \in \text{Names} \wedge w \text{OCCURS} \text{In}_n(p) \}
\]

where \( xOp\text{Occurs} \text{In}_n \) is a predicate expressing the occurrence frequency \((n)\) of a string \(x\) in a string \(y\).

As an example of characterizations based on the presentation of molecules, consider:

\[
\chi^p_{\mathbf{M}}(\text{"president is involved in marriage with politician"})
\]

= \{\{is, involved, in, marriage, politician, president, with\}\}

\[
\chi^p_{\mathbf{M}}(\text{"president was born in year"})
\]

= \{\{born, was, in, president, year\}\}

\[
\chi^p_{\mathbf{M}}(\text{"president 'J.F. Kennedy' was born in year 1917 and died at age 46"})
\]

= \{\{age, at, born, died, was, at, and, in, president, year\}\}

where \"president is involved in marriage with politician\" denotes the path expression (molecule) with verbalization president in marriage with politician. For the example, we have the following relevance:

\[
\chi^p_{\mathbf{M}}(\text{"president was born in year"}),
\]

\[
\chi^p_{\mathbf{M}}(\text{"president 'J.F. Kennedy' was born in year 1917, and died at age 46"})
\]

\[
= \{\{born, was, in, president, year\}\}
\]

\[
= \{\{age, at, born, died, was, at, and, in, president, year\}\}
\]

\[
\frac{5}{15} = 1/3 \approx 0.33
\]

A possible refinement for the weak characterization of molecules would be the introduction of stop and stemming lists, yielding a more refined base for characterization.

For weak characterizations, conceptual relevance makes less sense as it is a presentation based characterization rather than a contents based one.

6. QUERY BY CONSTRUCTION

The queries resulting from the explorative phase, supported by the query by navigation mechanism, are rather simple in that they always correspond to linear path expressions. This means that the resulting queries will never contain operations like \text{union}, \text{intersection}, \text{difference}, etc. Therefore we propose to integrate the query by navigation interface with a LISA-D structure editor.

When such a structure editor for queries is added the linear path expressions resulting from query by navigation can be combined into more complex ones, utilizing the expressive power of LISA-D to its fullest. For a complete discussion of all possible operations in LISA-D please refer to [14, 15, 37]. An example session of such an editor is depicted in Figure 21. The depicted query is a formulation in LISA-D of the request:

\text{List the parties which have a member who was a president, was born in Oregon, has model railroads as a hobby and was never vice president.}

The process of building a query by combining linear path expressions in such a structure editor, is referred to as \text{query by construction}. In this example we can see four linear path expressions:

\text{party which has as president, president who was born in the state, president who has the hobby, president who is the vice president of an administration}

![FIGURE 21. Query by construction.](image)
To create a more readable verbalization, the President object type has been removed from the last three linear paths. This is merely a verbalization issue though. Besides these four linear paths, this query contains the two operations: AND ALSO and BUT NOT and two constants: Oregon, Model railroads. The two operations correspond to multi-set intersection and multi-set difference respectively. It is quite possible to develop a direct manipulation interface [42] which allows users to manipulate these query components, while the system provides guidance based on the grammar of LISA-D. For other examples of graphical query interfaces, refer to [43,44].

The presentation of query results can be fully integrated with the query by navigation interface. A query result can be used as a starting point for a navigation session through the hyperbase. When evaluating a LISA-D query a (multi) set of binary tuples results. To this result the following set of molecules in the hyperbase can be associated:

\[
\text{Result}(Q) = \{A | \text{lnPop}(A,D)[Q]\}
\]

where \(D[Q]\) denotes the path expression associated to LISA-D query \(Q\). These results can then be presented by means of a (virtual) molecule as presented in Figure 22, from which instances can be selected for a further exploration of the hyperbase.

For a formal treatment of LISA-D, refer to [14] or [37]. The expressiveness of the LISA-D query language is high. As an example (taken from [37]), consider the use of recursive macros in the context of a conceptual schema for a two-person game as illustrated in Figure 23. In this domain the fact type may lead to describe how positions can be reached from one another. The unary fact type is a direct win gives all winning positions for the first player. The question now is to yield all positions from which the first player can win [45]. This is captured by the following macro:

\[
\text{Winning Positions IS is direct win UNION may lead to Position ALL IN Position may lead to Winning Positions}
\]

The above query is an example of a query that cannot be expressed as a so-called stratified query (see Figure 24, taken from [45]). Stratified queries can express all first-order queries and negation is allowed between the so-called strata. It has been shown, however, that stratified queries do not express all fixpoint queries, in particular, they have difficulty taking fixpoints over universal quantifiers, such as is needed in the above query [45,46].

The macromechanism of LISA-D allows for the specification of arbitrary fixpoint queries. There are, however, some relatively simple queries which cannot be expressed as fixpoint queries (due to lack of arithmetic operations). An example is the query Even, which determines whether the number of instances in a certain relation is even [45]. LISA-D provides the necessary basic arithmetic operators and this query can therefore be straightforwardly expressed:

\[
\text{EVEN R IS NUMBER OF(R) MOD 2 = 0}
\]

Whether every computable query can be expressed in LISA-D remains an issue for further research.

CONCLUSIONS AND DISCUSSION

In this paper a new approach to query formulation support in the context of information systems has been proposed. The query by navigation system can be used to formulate so-called linear path expressions in the conceptual query language LISA-D. This mechanism allows a user to navigate through the conceptual schema of an information system as well as the population of that system. The linear paths resulting from a query by navigation session can be combined into more complex
queries using a syntax directed editor for query by construction. The query language LISA-D, which has a high expressiveness, thus becomes an intuitive mechanism for the formulation of queries.

As a next step, the proposed querying system should actually be implemented, after which empirical testing can provide the feedback needed to make proper decisions on how to configure and tune the navigation system. The question that needs to be answered is, of course, to what extent such a system improves on the existing situations. Comparisons need to be made with plain SQL systems, Query by Example systems, and also with full natural language query approaches. The hypothesis that needs to be tested in each of these cases is: query by navigation helps users, who are not familiar with all aspects of the underlying domain, with the formulation of queries.

We can already identify some potential problems with the proposed use of query by navigation. First, query by navigation combined with query by construction indeed allows us to formulate any query we would like to formulate in a semi-natural language format. However, due to the restricted format of this natural language, users may develop a feeling of being forced into a straight-jacket (albeit a much more comfortable one than provided by SQL). This may in particular be the case for more experienced users. Full natural language query formulation tools, for example, put no real restrictions on the freedom of users to formulate queries. If this is found to be the case then the conclusion could be that query by navigation is a good tool to help users 'discover' the underlying conceptual schema, but that once they know 'what is out there', they should be able to use a query formulation interface that gives them more freedom. On the other hand, in the context of large conceptual schemata, knowing exactly what type of information is stored may become too hard a task. This could mean that users may still prefer to use query by navigation to get a feel of what information is at their disposal before formulating the actual query (either using query by construction or some natural language query tool).

In most real life applications, the size of conceptual schemas may quite well match that of wall paper. This leads to two practical problems when using query by navigation. First, users may still become lost in conceptual space when using a (simple) two-level query by navigation mechanism as presented in this article. Furthermore, simply starting a query by navigation session becomes a difficult task in itself. Every object type in the conceptual schema is a potential starting point for a navigation session. How can we ask a user to make a choice between 100 starting points? Whenever this becomes a problem we have the option to switch from the simple two-level approach to a truly multi-level stratified architecture for query by navigation. In conceptual modelling, abstraction techniques have been developed that allow us to take a conceptual schema and (automatically) derive multiple levels of abstractions from such a given schema [39, 40]. This would allow us to define a multi-level query by navigation space. Users would start at the highest level of abstraction where they are confronted with only a limited number of starting points. Theoretically, we can introduce so many abstraction layers that we will end up with only a handful of object types in the highest abstraction layer.

So in conclusion, after testing the basic idea of using query by navigation for the formulation of queries on databases, further testing is still needed to tune the system as is and investigate the need for multiple levels of abstractions to deal with large conceptual schemas.

ACKNOWLEDGEMENTS

We would like to thank the anonymous referees for their comments and suggestions, which have led to improvements in the original article.

The work reported in this paper has been funded in part by the Cooperative Research Centres Programme through the Department of the Prime Minister and Cabinet of Australia.

REFERENCES

Asymetrix (1994)


