

A GRAPHICAL QUERY LANGUAGE FOR HYPERTEXT DATABASE SYSTEMS

 Richard Gary Epstein
 Department of Statistics / Computer
 and Information Systems
 The George Washington University
 Washington, DC 20052

Abstract

This paper presents a semantic data model for the implementation of hypertext database systems. The data model provides a framework for improving the functionality of hypertext systems. An important part of the model is its user interface, called the "informatics calculus". Based upon the notion of a functional query language, this is a graphical query language with powerful facilities for expressing relationships and constructing new object classes from those provided within the database.

1. Introduction

There is a growing consensus regarding the functionality which must be built into the next generation of hypertext systems [1,2,3,4,5]. This paper discusses that functionality and proposes a framework for achieving it. That framework is the information resource data model [6,7]. "Informatics calculus" refers to the user interface which is an important part of the model. That interface takes the form of a graphical, functional programming language.

2. The next generation of hypertext

Halasz [8] lists seven capabilities which should be incorporated into the next generation of hypertext systems, capabilities which are absent from existing systems. The basic insight behind the information resource data model is that many of these capabilities can be achieved if one designs hypertext systems around formal data models. A similar notion is put forward in Akscyn et al. [9], who assert that user interfaces must be designed from the "inside-out", the inside being the data model and the outside being the user interface. The following paragraphs present those capabilities in Halasz's list which are most relevant to the current paper.

1. Search and query. Halasz notes that the browsing mechanism which is characteristic of hypertext is an inadequate mechanism for performing

searches. The informatics calculus query sublanguage was designed to enable users to specify the information they want based upon perhaps complex relationships within the database.

2. Composites. Halasz notes that hypertext systems lack a "composition mechanism", a means of "representing and dealing with groups of nodes and links as unique entities separate from their components" ([8], p. 843). The informatics calculus query sublanguage provides this capability in a very natural manner. Users can specify arbitrary collections of nodes, links and individual information fields within nodes using an algebra of functional forms. This algebra of functional forms treats links as data on a par with text, pictures, numbers and strings.

3. Computation in (over) Hypermedia Networks. Halasz notes that the database accessing mechanisms in hypertext are not integrated with computational mechanisms. The informatics calculus combines database access and database computations in one coherent framework. The informatics calculus expresses computations (such as finding the average of a set of numbers) and queries (such as retrieving all objects satisfying a given condition) in terms of the application of functional combinators to functions.

4. Extensibility and Tailorability. Halasz notes that the generic nature of the data model which is implicit in hypertext (nodes and links) does not permit users to easily tailor systems to their particular perceptions. The information resource model is closely related to the semantic data model of Hammer and McCleod [10] and can be classified as a semantic data model. Semantic data models are designed to capture user perceptions. Furthermore, the functional programming framework of the informatics calculus is inherently extensible, allowing implementors and users to introduce their own functions and applications into the algebra of functional forms.

5. Virtual Structures for Dealing with Changing Information. Halasz notes that existing hypertext systems are inflexible. Hypertext links are links between objects rather than classes of objects. This is due to the fact that there is no support for the database notion of "virtual structures". The class concept is an example of a "virtual structure". The information resource model is a traditional database data model in that the most significant structures are virtual. Particular objects are instances of their classes and the kinds of links between objects are determined by the classes to which they belong. The informatics calculus data update sublanguage takes advantage of this and allows the user to manipulate links en masse. Virtual structures also provide a basis for implementing database "views".

3. An object-oriented data model for hypertext

In this section we will sketch the basic properties of the information resource data model. The information resource data model is based upon Hammer and McCleod's semantic data model [10] and Shipman's functional data model [11]. These, in turn, have led to object-oriented models, such as the design model presented in Kroenke and Dolan [12] and the implementation model presented in Andrews and Harris [13].

This paper uses object-oriented terminology which is closely related to the terminology employed by Kroenke and Dolan. Their terminology is based upon the terminology software engineers employ in discussing "object-oriented design" [14]. This paper alters the Kroenke and Dolan terminology somewhat to bring it into conformance with the terms employed in object-oriented programming languages, such as Smalltalk [15].

Object-oriented concepts are closely related to the concepts of the functional data model. The functional data model provides the computation formalism behind the informatics calculus user interface. Table 1 provides a dictionary of object-oriented terms and the equivalent functional terms.

According to the information resource model (using the object-oriented terms) a database is a collection of object classes. Object classes are either scalar (STRING, NUMBER, DATE, etc.), hyper (DOCUMENT, PICTURE, VIDEO, etc.), or schema (classes defined by the database designer in a particular database schema). Each class consists of objects which are instantiations of that class. All objects in a class share a common set of properties. Properties can either be single-valued or multi-valued (yielding the "arity" of the property). The set of values that a given property can acquire is called its value class. Properties can

either be scalar (the value class is a scalar class), hyper (the value class is a hyper class), or schema (the value class is a schema class). Subclasses are classes which inherit all the properties of their superclasses, but may have additional properties of their own.

The next section will present a sequence of informatics calculus examples. These will be based upon the social studies database schema given in Table 2. This database schema includes three schema classes (STATES, PERSONS and INDUSTRIES) and one subclass (POLITICAL_PERSONS). The schema says, for example, that GOVERNOR is a single-valued schema property of the class STATES whose value class is POLITICAL_PERSONS. BIOGRAPHY is a single-valued hyper property of the class PERSONS whose value class is DOCUMENT. An arity of 1 indicates a single-valued property and an arity of M indicates a multi-valued property. An identifier is a property which uniquely identifies an object within a class.

Table 3 presents the functional data model schema which is equivalent to the object-oriented schema of Table 2. When we describe the semantics of informatics calculus expressions in terms of functional composition, Cartesian product and so forth, we are actually referring to the functions given in Table 3.

4. The informatics calculus query sublanguage

The informatics calculus was inspired by Buneman's functional query language (FQL) for the functional data model [16]. Buneman applied Backus' notion of an applicative language [17] to a database context. Buneman expressed database queries by combining functions (representing entity classes, attributes and predicates) using a small set of functional combinators (composition, Cartesian product, and iteration).

The informatics calculus extended the FQL formalism in several important regards. The informatics calculus expresses functional combinators using a visual metaphor, thus producing a language which is user-oriented. The underlying data model was extended to support multimedia, something which was not possible in Buneman's stream-oriented language. Intrinsic functions (called functionals and functors) were introduced to enable users to express important computations, such as computing averages and sorting objects. Finally, the formalism was extended to include a database update sublanguage.

Informatics calculus expressions take the form of a "mosaic", as shown in Figure 1. A mosaic consists of a collection of rectangular boxes called "tiles". Each tile contains a class, predicate, property, functional, functor or some other meaningful function. The

Table 1. Correspondence between functional and object-oriented terms.

object-oriented	functional
(object) class	entity class
object	entity
property	function (attribute)
value class	range of function (attribute)
scalar property	scalar attribute
schema property	entity-valued attribute
subclass w/inheritance	subclass w/inheritance

Table 2. A Social Studies information resource specified in terms of classes and their properties.

```

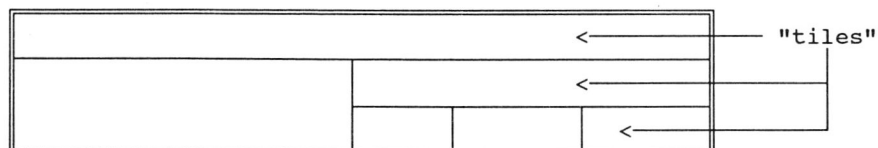
class STATES
identifier NAME
  property:          arity:          value class:
  NAME              1              STRING
  REGION            1              STRING
  POPULATION        1              NUMBER
  GOVERNOR          1              POLITICAL_PERSON
  INDUSTRIES        M              INDUSTRIES
  HISTORY           1              DOCUMENT

class INDUSTRIES
identifier NAME
  property:          arity:          value class:
  NAME              1              STRING
  HISTORY           1              DOCUMENT
  STATES            M              STATES

class PERSONS
identifier NAME
  property:          arity:          value class:
  NAME              1              STRING
  AGE               1              NUMBER
  SPOUSE            1              PERSONS

subclass POLITICAL_PERSONS of PERSONS
  property:          arity:          value class:
  PARTY             1              STRING
  
```

Figure 1. A mosaic consisting of six tiles.



spatial arrangement of tiles in a mosaic determines the manner in which functional combinators, such as composition and Cartesian product, are applied to the functions that the tiles represent.

We will now present a sequence of informatics calculus examples. The underlying formalism treats classes as functions which can be composed with their properties. Vertical juxtaposition denotes functional composition. Predicates are defined in such a way that functional composition of a class with a predicate yields a subset of the original class. Composing a class with a predicate corresponds to the select operation of the relational algebra. The composition of an class with a schema property behaves like the join operation of the relational algebra. The composition of an class with a scalar or hyper property is related to the project operation of the relational algebra.

Properties can be combined using Cartesian product. Cartesian product is denoted using horizontal juxtaposition. When the Cartesian product of properties is composed with a class, the result is similar to projection over two or more attributes in the relational data model. These basic capabilities define an important subset of the informatics calculus query sublanguage.

Example 1 presents an informatics calculus query which yields the names of the states in the northeast which have a Democratic governor as well as the names and ages of the governors. In this example, the class STATES is first composed with the predicate

REGION is "NORTHEAST"

This has the effect of selecting those states X which satisfy the constraint:

REGION(X) = "NORTHEAST"

The second predicate

PARTY of GOVERNOR is "DEMOCRATIC"

involves the derived property

PARTY of GOVERNOR

A derived property is a property which is computed using the properties which are native to the database schema. This second predicate has the effect of selecting those states X which satisfy the constraint:

PARTY (GOVERNOR(X)) = "DEMOCRATIC"

For each state X that satisfies the two constraints, the query of example 1 will yield the following values:

NAME(X), NAME(GOVERNOR(X)),
AGE(GOVERNOR(X))

The informatics calculus requires that all classes and properties be given in upper case and all reserved words of the language be given in lower case. The reserved word "of", for example, denotes functional composition (much like the symbol "o" in mathematics).

Informatics calculus queries can be assigned names. The query of example 1 has been given the name NE_DEM_STATES. The name is shown in a name tile at the upper left corner of the query.

Example 2 illustrates how hyper properties (pictures, documents, videos, etc.) are presented in the result part of an informatics calculus query. The result part of a query appears below the query when the query is evaluated. Hyper property values are presented as menu elements (shaded boxes in our example). A menu element represents the information implicit in the relevant document, picture or video. The user would point at the menu element in order to gain access to the indicated document.

The query of Example 2 asks for the names of states in the northeast with a population of at least 3 million. In addition, the query requests the names and histories of the industries in those states. The HISTORY property has the value class DOCUMENT, and thus, the histories of the various industries are displayed as menu elements.

Example 3 illustrates how the informatics calculus handles schema property values. In this case, the query asks for the names of states in the northeast that have a Democratic governor, and it also requests access to the governor objects as such, that is, without specifying particular properties of interest. The governors are displayed as menu elements in the result part of the query. These menu elements are labelled with the identifier specified for the class PERSONS (i.e., the person's name). The user can access all of the information relating to a particular governor by pointing to the menu element for that governor.

Example 4 illustrates how computations are incorporated into the informatics calculus functional programming framework. The reserved word "first" denotes an informatics calculus selecting functor. This functor acts upon a scalar property (which could be a derived property) and it converts that scalar property into a predicate which is true only for those objects which rank first in terms of the given property. For example, the functor

first / POPULATION

applied to a set of states, would select that state which ranks first by population (perhaps more than one state if there is a tie). The functor

first / AGE of SPOUSE of GOVERNOR

applied to a set of states, would select that state (in the given set) whose governor has the oldest spouse, that is, the state X which ranks first according to the value of the function







AGE(SPOUSE(GOVERNOR(X))).

Consequently, the query of example 4 first selects those states in the northeast that have a population above 4 million and a Democratic governor. Then, the query selects that state (among those remaining) whose governor has the oldest spouse. The query then requests the name of that state and the name of the governor of that state.

Example 1. Accessing the names of states in the northeast that have a Democratic governor, as well as the names and ages of the governors.

NE_DEM_STATES		
STATES		
PARTY of GOVERNOR is "DEMOCRATIC"		
REGION is "NORTHEAST"		
NAME	GOVERNOR	
	NAME	AGE

Example 2. Accessing the names of states in the northeast with a population above 3,000,000, as well as the names and histories of the industries for those states.

NE_BIG_STATES		
STATES		
REGION is "NORTHEAST"		
POPULATION > 3,000,000		
NAME	INDUSTRIES	
	NAME	HISTORY
CONNECTICUT	INSURANCE	document 
	FIREARMS	document 
MASSACHUSETTS	HIGH TECHNOLOGY	document 
	SOUVLAKI	document 
NEW YORK	BASEBALL	document 
	FINANCE	document 

Our final informatics calculus example illustrates that one query can refer to another. The informatics calculus query `BIG_NE_DEM_STATES` of Example 5 gives the names, populations and names of governors for states in the northeast that have Democratic governors, whose list of industries includes the insurance industry and whose population is above the national average population for states. The predicate

`POPULATION > AVE_POP`

refers to a second query, whose name is `AVE_POP`. `AVE_POP` is declared with the type restriction `1(NUMBER)`, indicating that it computes a single number and can thus be used on the right hand side of a predicate whose left hand side denotes a single-valued property whose value class is a `NUMBER`. Also note that the predicate `NAME of INDUSTRIES` includes

`"INSURANCE"`

is true if the set `NAME` of `INDUSTRIES` for a given state includes the string `"INSURANCE"`.

5. Different perspectives on informatics calculus

In this section we will view several alternative ways of viewing the informatics calculus.

1. **A menu generating system.** The informatics calculus allows the user to generate menus of nodes, data and links in a hypertext database which is designed around the information resource model. These menus provide gateways into the hypertext system, gateways which help the user to overcome the cognitive overhead and disorientation problems described by Conklin [1]. In other words, these user-generated menus provide the user a means

Example 3. Gaining access to all Northeast Democratic governors without specifying particular properties of governors.

NE_DEM_GOVs	
STATES	
PARTY of GOVERNOR is "DEMOCRATIC"	
REGION is "NORTHEAST"	
NAME	GOVERNOR
CONNECTICUT	O'NEILL
MASSACHUSETTS	DUKAKIS
NEW YORK	CUOMO
VERMONT	KUNIN

Example 4. Finding out which northeast Democratic governor in a state with a population above 4,000,000 has the oldest spouse. Give the name of that governor and the name of his / her state.

NE_DEM_GOV_OLD_SPOUSE	
STATES	
PARTY of GOVERNOR is "DEMOCRATIC"	
REGION is "NORTHEAST"	
POPULATION > 4,000,000	
first / AGE of SPOUSE of GOVERNOR	
NAME	GOVERNOR
	NAME

of managing the complexity of connections in a hypertext system.

2. A menu-driven system. In terms of the basic interaction styles for interactive systems [17,18], the informatics calculus interface is designed as a menu-driven system. The informatics calculus user constructs mosaics by making choices in a system of menus. In this context, the mosaics are not just queries. They provide significant visual feedback to the user concerning the choices made thus far. The mosaics not only help the user to manage complexity (the query language aspect), they also provide temporal and spatial orientation with respect to the information retrieval task (the visual feedback aspect).

3. Informatics calculus as a specification language. The informatics calculus provides a framework for formally specifying the construction of new classes

from the collection of classes contained in a database schema. The functional combinators (Cartesian product and functional composition) correspond to snipping and pasting operations with respect to classes. Thus, the informatics calculus plays a role in object-oriented databases similar to the role played by the relational algebra in relational database.

For example, the query of Example 6 can be viewed as specifying a new class, `BIG_NE_DEM_STATES`, which contains information extracted from the `STATES` and `POLITICAL_PERSONS` classes.

6. Properties of a tiling editor

The informatics calculus user would construct mosaics using a "tiling editor". The tiling editor would be a menu-driven system which would provide the user with essentially three kinds of menus:

1. Tiling operator menus. These would allow the user to choose from a small set

Example 5. A query (BIG_NE_DEM_STATES) which utilizes the value of another query (AVE_POP).

AVE_POP: 1(NUMBER)
STATES
POPULATION
average

BIG_NE_DEM_STATES		
STATES		
PARTY of GOVERNOR is "DEMOCRATIC"		
NAME of INDUSTRIES includes "INSURANCE"		
REGION is "NORTHEAST"		
POPULATION > AVE_POP		
NAME	POPULATION	GOVERNOR
		NAME

of fundamental tiling operators.

2. Function menus. These would provide users with a menu of functions, functors and functionals which could be inserted into a particular tile depending upon the context.

3. Help menus. These would provide users with various kinds of help, including roadmaps of the underlying database schema.

The existence of a small collection of fundamental tiling operators is essential to the success of the informatics calculus as a viable user interface. Preliminary work on the design of a tiling editor indicates that there are indeed only a few fundamental operators, few enough so that they could be conveniently expressed either in a function key or mouse-driven interface. Table 3 provides a list of partial list of fundamental tiling operators and how they might be assigned to function and cursor keys.

7. Summary

The information resource model represents a proposal for accomplishing a melding of hypertext and state of the art database technologies. The informatics calculus, the query language of the information resource model, illustrates how database query languages can help hypertext users manage complexity. The informatics calculus is a menu-driven system, where the informatics calculus mosaics give the user visual feedback concerning the current state of the information retrieval task being formulated.

References

1. Conklin, J. (1987) "Hypertext: An Introduction and Survey", IEEE COMPUTER, 20(9), pp. 17-41.
2. Yankelovich, N. et al. (1988) "Intermedia: The Concept and the Construction of a Seamless Information Environment". IEEE Computer, 21(1), pp. 81-96.
3. Marchionini, G. and Shnediderman, B. (1988) "Finding Facts vs. Browsing Knowledge in Hypertext Systems", IEEE Computer, 21(1), pp. 70-80.
4. Yankelovich, N., Meyrowitz, N. and van Dam, A., (1985) "Reading and Writing the Electronic Book", IEEE Computer, 18(10), pp. 15-29.
5. Wegner, P., (1984) "Capital-Intensive Software Technology", IEEE Software, 1(3), pp. 7-45. [Especially part 3 on knowledge engineering.]
6. Epstein, R. G. (1988) "Informatics Calculus: A Graphical, Functional Query Language for Information Resource Systems", doctoral dissertation, Department of Computer and Information Sciences, Temple University.
7. Epstein, R. G. (1988) "The Informatics Calculus: A Graphical, Functional Query Language for Information Resources", RIAO 88 Conference Proceedings, pp. 667-690, March 1988, Cambridge, MA.

8. Halasz, F. (1988) "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems", Communications of the ACM, 31(7), pp. 836-855.
9. Akscyn, R., Yoder, E., and McCracken, D. (1988) "The Data Model is the Heart of Interface Design", in CHI '88 Conference Proceedings, pp. 115-120, May 1988, Washington, D. C.
10. Hammer, M. and McCleod, D. (1981) "Database Description with SDM: A Semantic Data Model", ACM Transactions on Database Systems, 6(3), pp. 351-386.
11. Shipman, D. (1981) "The Functional Data Model and the Data Language DAPLEX", ACM Transactions on Database Systems, 6(1), pp. 140-173.
12. Kroenke, D. and Dolan, K. (1988) Database Processing, SRA, Chicago.
13. Andrews, T. and Harris, C. (1987) "Combining Language and Database Advances in an Object-Oriented Development Environment", OOPSLA '86.
14. Abbot, R. (1983) "Program design by informal English descriptions", Communications of the ACM, 26(11), pp. 882-894.
15. Goldberg, A. and Robson, D. (1983) Smalltalk-80: The Language and its Implementation, Addison-Wesley, Reading, MA.
16. Buneman, P., Frankel, R., & Nikhil, R. (1982) "An Implementation Technique for Database Query Languages", ACM Transactions on Database Systems, 7(2), pp. 164-186.
17. Backus, J. (1978) "Can programming be liberated from the von Neumann style? A functional style and its algebra of programs", Communications of the ACM, 21(8), pp. 613-641.
18. Foley, J. D., Wallace, V. L., and Chan, P., (1984) "The Human Factors of Computer Graphics Interaction Techniques", IEEE Computer Graphics and Applications, 4(11), pp. 13-48.
19. Shneiderman, B. (1987) Designing the User Interface, Addison-Wesley, Reading, MA.

Table 3. Some fundamental tiling operations.

Operator:	Function or cursor key:
Backtrack (undo previous operation)	F1
Split current tile vertically	F2
Split current tile horizontally	F3
Change current tile to neighbor	cursor keys
Remove current tile boundary	F4, followed by cursor key
"Color in" current	input string
Evaluate	F7
Retrieve mosaic from catalog	F8, followed by mosaic name
Save mosaic in catalog	F9, followed by mosaic name