A Universal Query Interface for Heterogeneous Distributed Digital Libraries

Norio Katayama, Masanori Sugimoto, and Jun Adachi
NACSIS (National Center for Science Information Systems)

Abstract

Today, various kinds of digital libraries are accessible on the WWW (World Wide Web). The interoperability among those systems is one of the major topics of the digital library community. This paper presents a HyNeSS model and its query interface for that purpose. The HyNeSS model integrates the entity relationship model and the predicate logic, and provides a universal and formal expression for network-oriented data. We have developed a prototype system of the HyNeSS database system. The query interface is universal in the sense that it does not depend to any particular database schema. It loads the schema information from a remote HyNeSS database system and adapts itself to the given schema. On the submission of a query, the composed expression is converted into a declarative query expression based on the predicate logic. It provides a universal expression to retrieve and identify objects in heterogeneous distributed environment.

1 Introduction

This paper presents a data model, HyNeSS, and its universal query interface for heterogeneous distributed digital libraries. It is universal in the sense that it provides a general expression for queries and does not depend on a particular database schema. The major property of our approach is an integration of the entity relationship model and the predicate logic. The query interface employs an expression based on the entity relationship model. The composed query expression takes the form of the entity relationship model. Then, it is converted into an expression based on the predicate logic on submission and the logic-based expression is exchanged between a client and a server. For the sake of the declarative semantics of the predicate logic expression, this method achieves the effective scheme to retrieve and identify objects in heterogeneous distributed digital libraries.

This paper is organized as follows. Section 2 describes the background of this study and Section 3 provides the system overview of the developed prototype system. Section 4 and 5 presents the HyNeSS model and its query interface. Section 6 discusses the properties of our approach and conclusion is contained in Section 7.

2 Background

Recently, experimental and operational digital libraries are accessible on the WWW (World Wide Web). For example, the digital libraries developed by Cornell University[6] and University of Michigan[2] are open to the public. In addition, index servers for WWW resources are provided by Yahoo, WWW Worm, Lycos, etc. Now, various and numerous digital repositories are available on the WWW.

To use these digital repositories seamlessly, the interoperability of digital libraries is one of the major topics of the digital library community. In the traditional sense, it is the problem of the integration of heterogeneous distributed databases, which is already discussed in the database literatures[7]. The key is how to create a federated schema and a common user interface. Although that problem has been discussed in terms of the conventional database systems, e.g., hierarchical, network, and relational database systems, the conventional approach is not directly applicable to digital libraries for the sake of the mismatch in data organization. Data stored in digital libraries have the form of hypermedia and hypertexts, i.e., rather network-oriented than record-oriented. Therefore, a new data model and a query interface are required which is suitable for network-oriented data.

There are three levels of the interoperability: not-coupled, loosely-coupled, and tightly-coupled. These levels can be distinguished in terms of a database schema and a query interface as shown in Figure 1. Not-coupled databases use different schemata and difference query interfaces. Users need to know every schema and every query interface to use them. Loosely-coupled databases share their query interface. Users can use the common syntax to express queries. Tightly-coupled databases share their database schema as well as their query interface. The shared schema may be a federated one, i.e., it may be a union of schemata of component databases. Users need not know the existence of component databases and can use it as if it is a single centralized database.

	Database schema	Query interface
Tightly-coupled	shared (federated)	shared
Loosely-coupled	not shared	shared
Not-coupled	not shared	not shared

Figure 1: The three levels of the interoperability

In the following sections, we present a data model HyNeSS and its query interface for the basis of a federated database schema and a universal query interface for heterogeneous distributed digital libraries.

3 System overview

Figure 2 illustrates the outline of our prototype system. The server has four components. The database is managed by a database management system HyNeSS (Hyper-Network Storage System) [5]. The httpd stores the schema information of the database and CGI (common gateway interface) programs which implement the access to the HyNeSS database. On the other hand, the client is an ordinary WWW browser which supports Java applets. The query interface is implemented with the Java language and executed by the WWW browser.

The sequence of a query processing is as follows:

(0) The Java applet of the query interface starts in the WWW browser.

(1) The applet loads the schema information from the remote server.

(2) The applet starts the universal query interface based on the entity relationship model.

(3) The applet sends the composed query in the form of an expression based on the predicate logic.

(4) The httpd throws the query to the HyNeSS database via CGI program and the HyNeSS answers the query.

The advantages of this approach can be summarized as follows:

1. The interface, expression, and applet are independent of a database schema.
   - → This sequence can be applied for any database as long as its schema information is provided.
   - → Users can use a single same interface for any databases.
2. Both the query interface and the query expression have declarative semantics.
   - → Users only need to express the query declaratively without considering how it will be processed by the server.
   - → Because both the entity relationship model and the predicate logic have declarative semantics, they can be directly converted into each other.

4 HyNeSS model

4.1 Outline

The HyNeSS stands for the Hyper-Network Storage System and is designed to model network-oriented data like hypermedia and hypertexts[5]. The HyNeSS model is formerly called the S-object model which is presented in [3, 4].

The HyNeSS model integrates the entity relationship model and the predicate logic. In the conceptual view, the HyNeSS model is an extended version of the entity relationship model[1]. It models the real world enterprise by entities, links, classes, and relationships. Entities and links are lowest level objects of a database and compose a network structure similar to semantic networks. Classes and relationships are meta level objects of entities and links, respectively. A class is a set of entities which is categorized by the semantic role of entities, while a relationship is a set of links categorized by the semantic role of links.

On the other hand, in the descriptive view, the HyNeSS model is based on the predicate logic. It has one-to-one mapping between the conceptual view and the descriptive one. An entity corresponds to a constant, a link to an atom, a class to a unary predicate, and a relationship to a n-ary predicate. One of the advantages of having the logic-based descriptive view is that it enables to define deductive rules for the conceptual view. The HyNeSS database management system has the deduction mechanism based on the function-free definite program which incorporates the novel mechanism to handle the procedural derivation[4]. We do not mention its detail here because it is beyond the scope of this paper.

The major property of the HyNeSS model is its declarative semantics. In the conceptual view, it employs the plain network expression, while it employs the function-free definite program in the descriptive view. The procedural semantics, e.g., character string matchings, image processing, mathematical computations, are encapsulated into predicates[3, 4]. With its declarative semantics, the HyNeSS model enables users to express what he/she wants to retrieve declaratively and enables distributed sites to communicate with each other in a declarative manner. Hence, the HyNeSS model provides the basis of a universal query interface and a universal expression for heterogeneous distributed digital libraries.

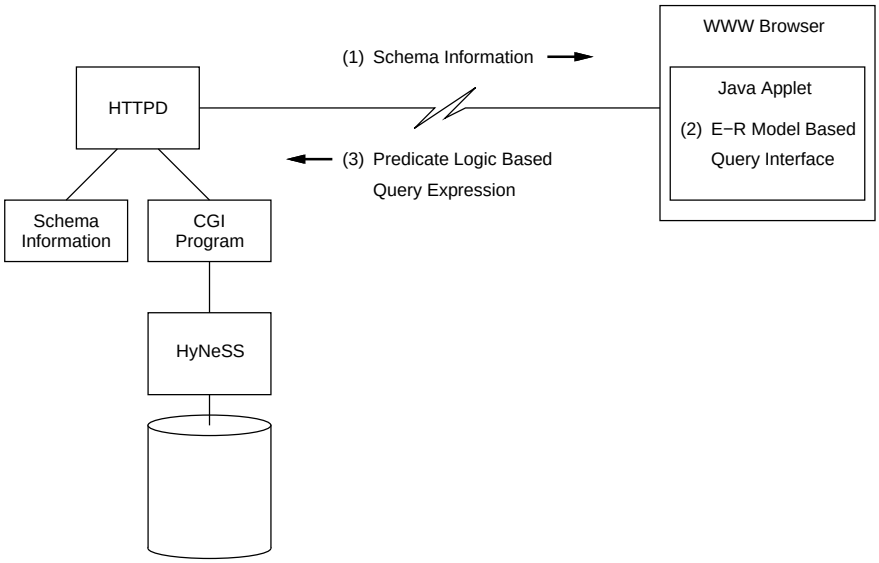


Figure 2: System overview

4.2 Definition

1. HyNeSS model
   The HyNeSS model consists of six types of components (E, C, T, L, R, D). E is a set of entities. C is a set of classes. T is a set of types. L is a set of links among entities. R is a set of relationships. D is a set of domain constraints.
2. Entities
   An entity is any substance, either conceptual or physical one, which needs to be distinguished within a database. Each entity has a unique identifier to distinguish itself from others. The current implementation of the HyNeSS database management system uses the hierarchical name space similar to the one of UNIX file systems. For example, a society IPSJ has an identifier `/society/ipsj` and an issue of the IPSJ transaction whose volume and number are 35 and 1 has an identifier `/issue/society/ipsj/tipsj/35/01`. The role of entities are to distinguish substances within a database. An entity has no attribute, differently from the object-oriented data model. Its attributes or structures are represented by links and relationships.
3. Classes
   Classes are sets of entities which are categorized by the semantic role of entities. The role of classes is to distinguish sets of entities and to be the basis of the domain constraints. For example, the following classes are defined for the prototype digital library: societies, journals, issues, pages, images, articles, and authors.
   A class itself decides neither attributes nor structures of entities, differently from the object-oriented data model. The constraints for entities are only defined by domain constraints of relationships.
4. Types
   A type is a kind of classes and represents a particular type of atomic data, e.g, strings, integers, etc. A type is defined with three components (T_C, T_E, T_D). T_C is a class identifier of the type, e.g., `/class/string`, `/class/integer`, etc. T_E is an encoder which converts each atomic data into its unique identifier, while T_D is a decoder which converts an identifier to its corresponding data. By providing such an encoder and a decoder, every atomic data owns its unique identifier and is treated as one of entities within a database. For example, the current prototype system encodes the character string "Digital Library" into an identifier `/string/Digital+Library` and the integer value "1996" into `/integer/1996`. Thus, the HyNeSS model assigns a unique identifier to each atomic data and treats them as entities. This approach can be regarded as a variant of a tagged architecture in which every binary data carries a tag which represents its data type.
5. Links
   A link is a tuple of entities which represents a relationship among entities. In the traditional notation, a link is expressed with the ordered list of entities with predicates. For example, the relationship between an author `/author/A` and his article `/article/A` is expressed as follows:

   ```
   Author-Article ( /author/A, /article/A )
   ```

   On the other hand, the HyNeSS model uses an order-independent notation to emphasize the symmetrical characteristics of terms. For example, the

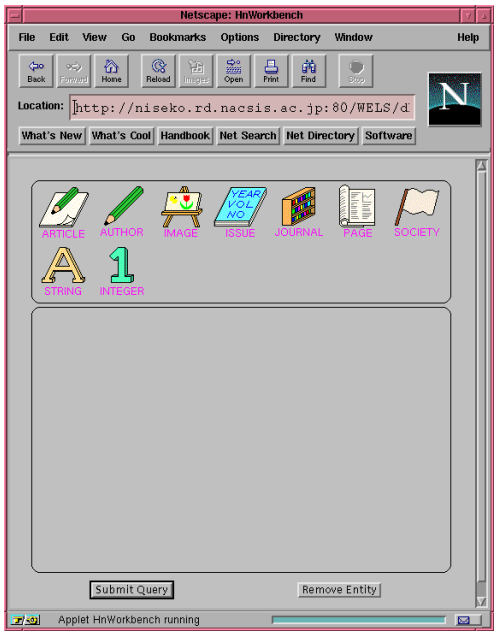


Figure 3: Universal query interface

relationship above is expressed as follows:

```
[ Author=/author/A, Article=/article/A ]
```

Terms are composed of label and value pairs and the predicate of a link is determined by the combination of labels. For the example above, the predicate Author-Article is determined by the combination of labels, Author and Article.

6. Relationships

   Relationships are meta objects of links. A relationship decides the predicate and the number of terms of links, i.e., links which belong to the same relationship must have the same predicate and the same number of terms. For example, the prototype system defines the following relationships for articles:

```
Article-Type                Article-Issue
Article-EnglishAbstract     Article-Author
Article-JapaneseAbstract    Article-FirstPage
Article-EnglishTitle        Article-LastPage
Article-JapaneseTitle       Article-Language
```

7. Domain Constraints

   A domain constraint restricts the values of a term of a relationship. A domain constraint is defined with three types of components (D_R, D_T, D_C). The combination of D_R and D_T specifies a term of a relationship, and D_C specifies a set of classes which is applicable to the term. In other words, it specifies that the values of the term D_T of the relationship D_R must be an instance of one of the classes included in D_C. The role of domain constraints is to specify the scope of relationships and prevents users from creating nonsense links by mistake.

5 Universal query interface

5.1 Overview

Figure 3 shows the initial screen of the universal query interface. It consists of two parts: a palette and a canvas. The palette is located above and contains icons of classes. The contents of the palette is determined by the schema information supplied by each database. The canvas below is a workbench of query expressions. On the canvas, users construct queries in the form of a network structure based on the entity relationship model.

The primitives of the query construction are as follows:

(1) Select a class icon from the palette, then drag and drop it on the canvas (Figure 4-(a)).

(2) Let the interface display the list of relationships applicable to a particular icon by double-clicking it on the canvas, then select one of the relationships to produce a link on the canvas (Figure 4-(b) and 4-(c)).

(3) Let the interface indicate the icons applicable to link ends (Figure 4-(d)).

(4) Let the interface pop up a dialog window by double-clicking an icon with pressing the shift key and then assign a constant value to the icon (Figure 4-(j)).

(5) Select icons to be displayed as the query result by clicking them on the canvas (Figure 4-(l)).

The advantages of the universal query interface are as follows:

1. It is independent of a database schema.

   It adapts itself to each database schema on the following aspects:

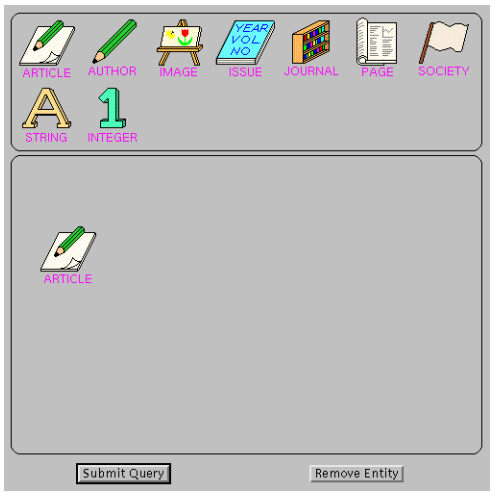
   - It decides the contents of its palette according to the loaded schema information.
   - It displays the list of applicable relationships according to the domain constraints.
   - It indicates icons applicable to a particular link end according to the domain constraints.

   By this adaptiveness, the query interface can be applied for any database as long as its schema information is provided.

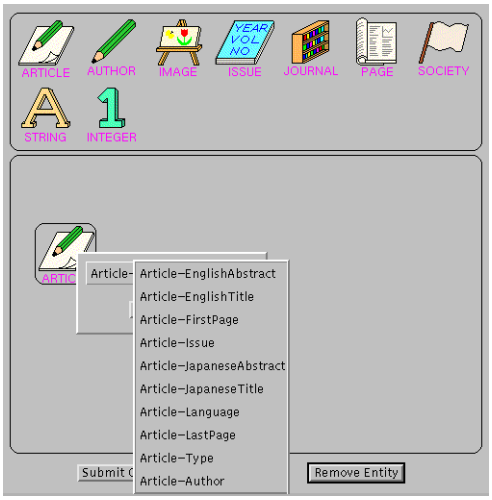
2. It is self-explanatory with respect to a database schema.

   Users can learn a database schema from the query interface itself. They do not need to consult the document describing the database schema. The interface explains a database schema to users by arranging class icons on the palette, displaying the list of applicable relationships, and indicating icons applicable to link ends.

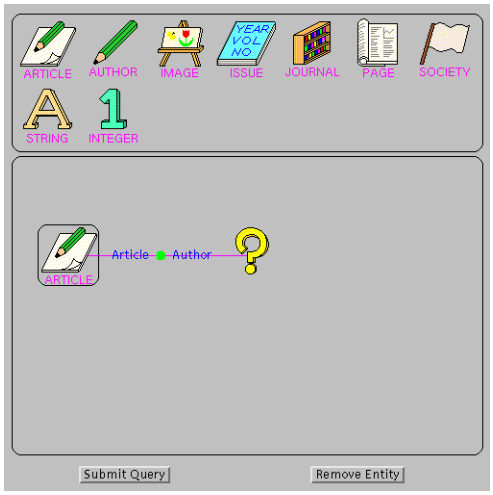
3. The constructed expression has declarative seman-



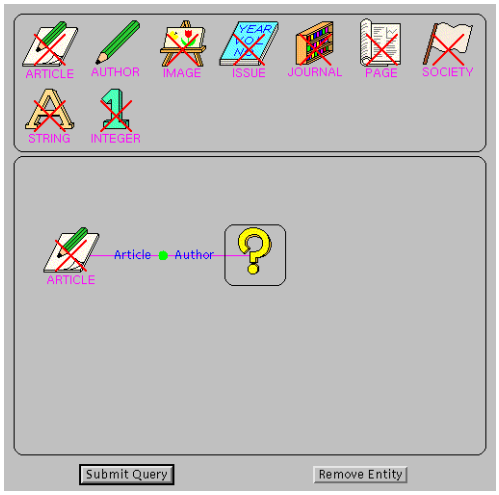
(a) An icon 'article' is dropped on the canvas.



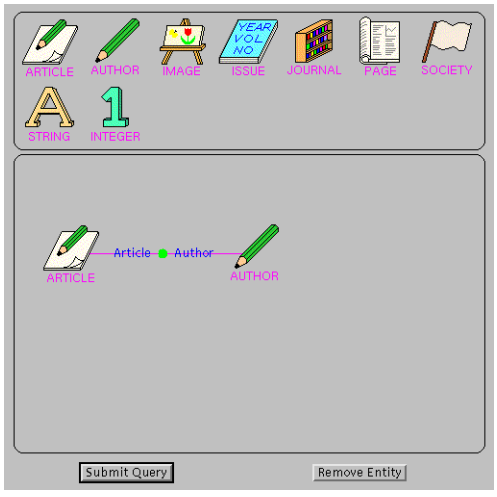
(b) The list of applicable relationships are displayed.



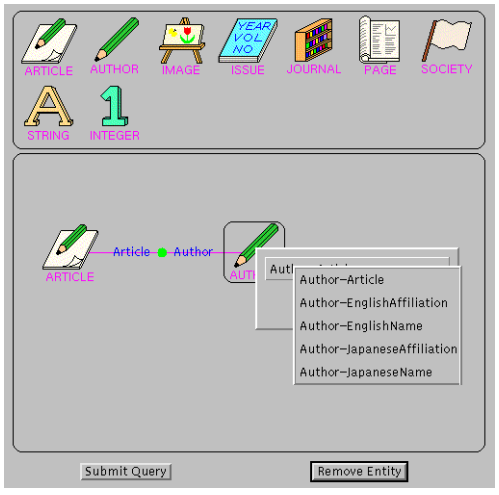
(c) The relationship 'Article-Author' is selected.



(d) The applicable icons to the link end is displayed.

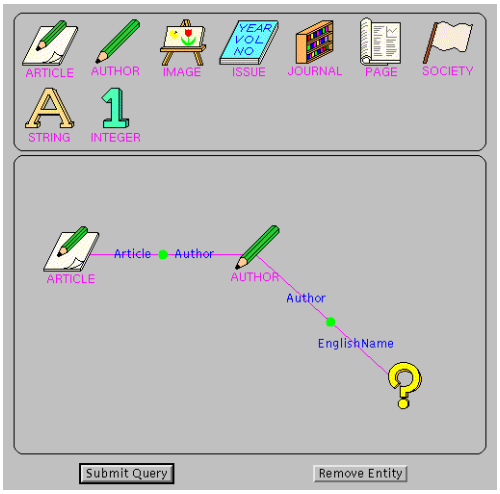


(e) The icon 'author' is selected for the domain.

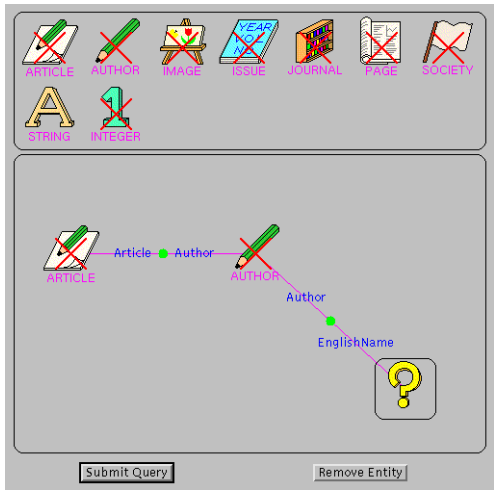


(f) The list of applicable relationships is displayed.

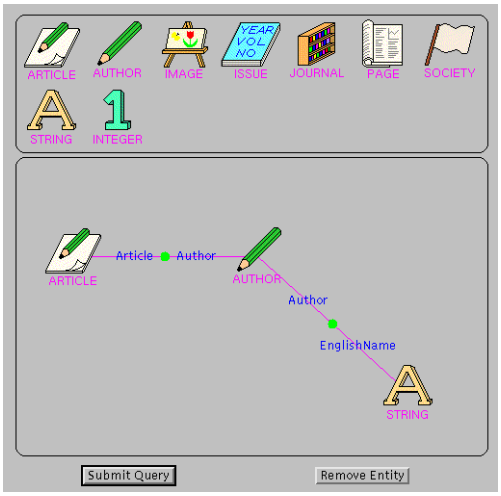
Figure 4: Example of query construction



(g) 'Author-EnglishName' is selected.

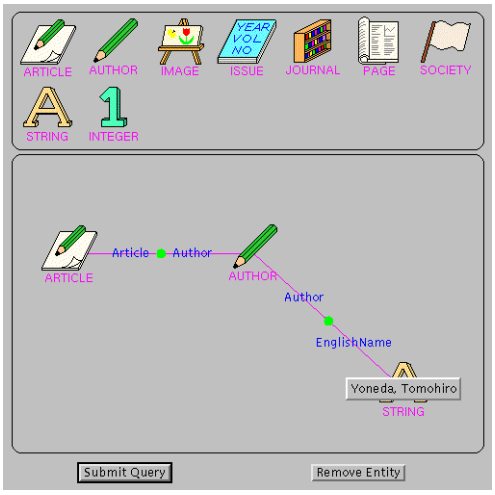


(h) The domain of 'EnglishName' is displayed.

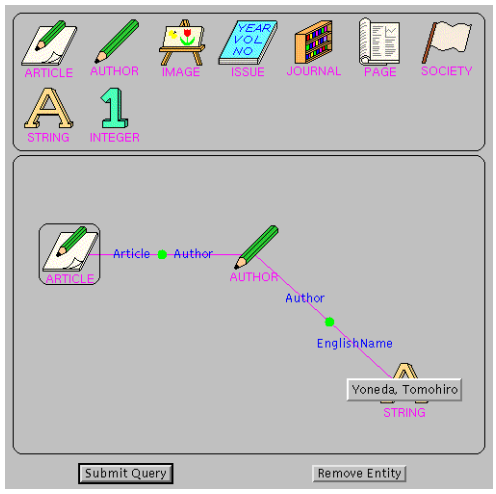


(i) The class 'string' is selected for the domain.

(j) A constant is being assigned to the 'string'.

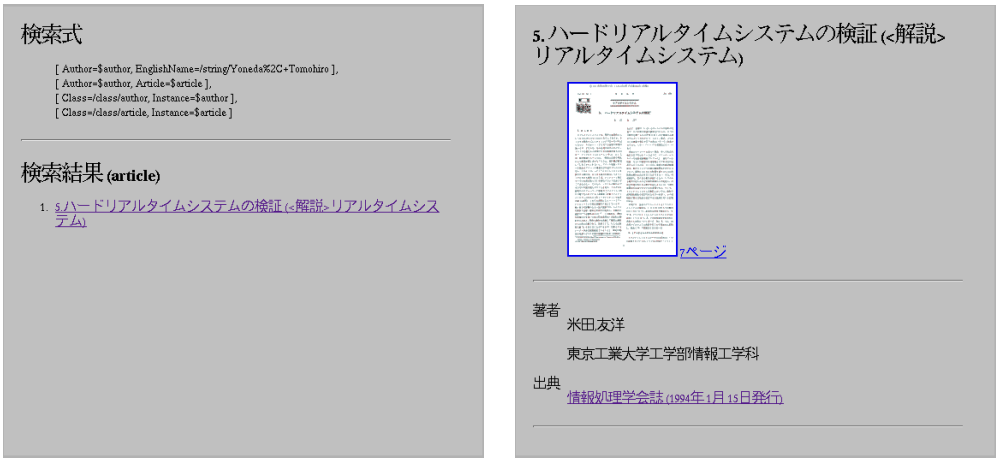


(k) 'Yoneda Tomohiro' is assigned to the 'string'.



(l) The icon 'article' is selected for the result.

Figure 4: Example of query composition (cont'd)



(a) List of query result (b) resultant article

Figure 5: Example of query result

tics.

The conceptual view based on the entity relationship model enables users to express a query declaratively without considering how it will be processed by the server.

5.2 Example of query construction

1. A user selects what to retrieve from the class icons on the palette. For example, the class 'article' is selected and dropped on the canvas in Figure 4-(a).
2. The user produces a link to specify a search condition for the selected icon. By double-clicking the icon, the list of applicable relationships is displayed as shown in Figure 4-(b). The query interface constructs the list according to the schema information supplied by the database.
3. If a relationship is selected from the list, a link of the relationship is produced on the canvas. However, the other end of the link is a question mark as shown in Figure 4-(c) because the icon of that end is not specified by the user yet. By double-clicking the question mark, the applicable icons to that end are indicated by the query interface as shown in Figure 4-(d) where the inapplicable icons are marked with crosses. The query interface determines applicability of icons according to domain constraints of the relationship.
4. By selecting icons for link ends, a link between entities is completed as shown in Figure 4-(e). The user repeats the above sequence until the query expression represents what he/she wants to retrieve. For example, in Figure 4-(f) to 4-(i), a link of the relationship 'Author-EnglishName' is added to the icon 'author'.
5. The user may assign a constant to icons as shown in Figure 4-(j) where a character string "Yoneda, Tomohiro" is assigned to the icon 'string'.
6. Finally, the user selects the icon to be displayed as the query result. Icons can be selected by clicking them on the canvas. The query interface marks selected icons by rectangles. For example, in Figure 4-(l), the icon 'article' is selected. This expression specifies to display articles written by an author whose English name is "Yoneda, Tomohiro".

   Then the user submits the query to the server by pressing the button "submit query". An example of the query result is shown in Figure 5. Figure 5-(a) shows the list of articles matched to the search condition. By navigating through the anchor associated to the items of the list, the user can view the resultant articles as shown in Figure 5-(b).

6 Discussion

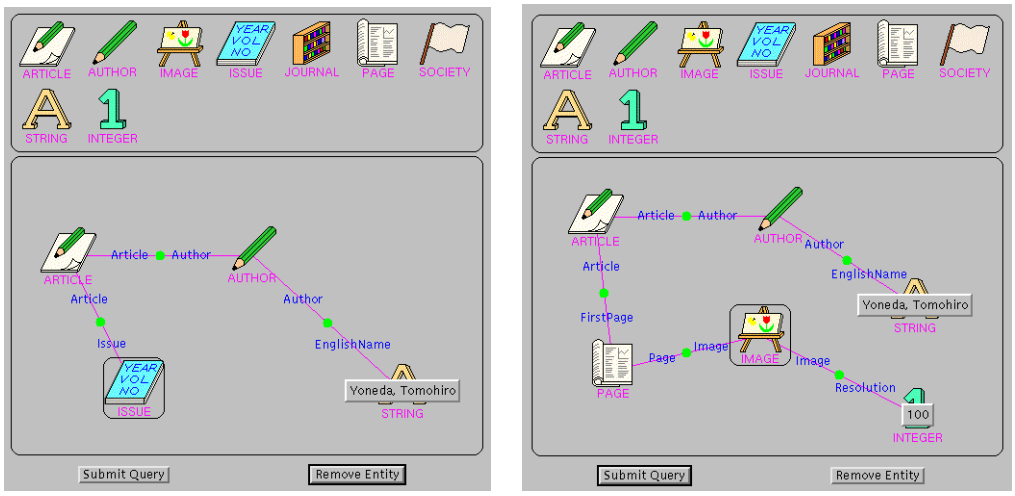
6.1 Effectiveness of declarative expression

The major property of our approach is that we employ declarative expressions for the query interface. Declarative expressions are significantly effective when queries are more complicated. Figure 6-(a) and (b) represents query expressions which specifies the following search conditions respectively:

(a) Display issues which contain articles written by an author whose English name is "Yoneda, Tomohiro".

(b) Display images of which resolutions are 100 DPI and which are the first pages of articles written by an author whose English name is "Yoneda, Tomohiro".

It is much more difficult to express these queries in the form of query commands such as SQL. It is even difficult to write them in natural languages. Declarative expressions are straight forward and easy to be



(a) (b)

Figure 6: Examples of more complicated queries

understood.

6.2 Applicability to other applications

Essentially, this approach is not restricted to digital library applications. It can be applied to general heterogeneous distributed databases. However, we regard that it is most effective for digital libraries because the HyNeSS model is designed to model network-oriented data such as hypermedia and hypertexts.

6.3 Future work

The desirable capabilities of the current implementation are as follows:

- Nested queries which enable the result of one query to be used in another.
- Disjunctive expressions.
- Negative expressions.

We plan to incorporate these capabilities to capture more semantics in query expressions.

7 Conclusion

In this paper, we presented the HyNeSS model and its query interface which provide the basis for heterogeneous distributed digital libraries. The HyNeSS model provides the universal modeling scheme for network-oriented data like hypermedia and hypertexts. The capability of the HyNeSS model is demonstrated by developing its query interface. It is schema-independent, self-explanatory, and declarative. Although its application field is not restricted to digital libraries, it is especially suitable for heterogeneous distributed digital libraries because the HyNeSS model is designed to model the contents of digital libraries such as hypermedia and hypertexts.

Acknowledgments

This research was partly supported by JSPS (Japan Society for the Promotion of Science). The authors would like to express their gratitude, for the helpful comments, to the members of the joint research program, "A Study on a Digital Multimedia Library for Supporting Creative Research," sponsored by JSPS.

References

[1] Chen, P., P.-S., "The Entity-Relationship Model – Toward a Unified View of Data," *ACM Trans. Database Systems* **1**, 1 (Mar 1976) 9–36.

[2] Crum, L., "University of Michigan Digital Library Project," *Communications of the ACM* **38**, 4 (Apr 1995) 63–64, `http://http2.sils.umich.edu/UMDL/` .

[3] Katayama, N., Takasu, A., and Adachi, J., "A Database with an Explicit Semantic Representation," *The Seventh International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems* (May 1994) 323–332.

[4] Katayama, N., Takasu, A., and Adachi, J., "A Logic-Based Approach for Managing Structured Document Data," *Proc. of the International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)* (Oct 1994) 245–252.

[5] Katayama, N., Takasu, A., and Adachi, J., "Design and Implementation of a Storage System based on Hypermedia Network," *IPSJ SIG Notes* 95-DBS-104 (Jul 1995) 57–64 (written in Japanese).

[6] Lagoze, C. and Davis, J., R., "Dienst: An Architecture for Distributed Document Libraries," *Communications of the ACM* **38**, 4 (Apr 1995) 47, `http://cs-tr.cs.cornell.edu/` .

[7] Sheth, A., P., Larson, J., A., "Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases," *ACM Computing Surveys* **22**, 3 (Sep 1990) 183–236.