Hubble: An Advanced Dynamic Folder System for XML

Ning Li Joshua Hui Hui-I Hsiao Kevin Beyer

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120 USA

{ningli, jhui, hhsiao, kbeyer}@almaden.ibm.com

ABSTRACT

Organizing large document collections for finding information easily and quickly has always been an important user requirement. This paper describes a flexible and powerful dynamic folder technology, called ***Hubble***, which exploits XML semantics to precisely categorize XML documents into categories or folders.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information filtering, Query formulation*, *Retrieval models.*

General Terms

Algorithm, Management, Performance, Design.

Keywords

XML, dynamic folder, categorization, content navigation

1. Introduction

With the fast advancement in CPU and disk technologies, significant amount of data and files are created and stored in computer systems nowadays. As a result, users and institutions are facing a serious challenge in managing and organizing their documents and files such that information can be found and retrieved easily and quickly. There are several known technologies for organizing and/or categorizing documents and web pages. The most familiar ones include folder and directory structures [1][2][3] for organizing files and categorization and classification technologies for grouping web pages and documents. Existing folder technologies place documents into folders either manually or automatically but based only on simple search criteria. The categorization and classification technologies automate the placement and grouping of documents and pages, but they are imprecise.

In the last few years, XML has become the de-facto standard for content publishing and data exchange. The proliferation of XML documents and data has created new challenges and opportunities for managing document collections. Since XML documents are self describing, it is now possible to automatically categorize XML documents precisely, according to their content. In addition, with the availability of the standard XML query languages, XPath and XQuery [4], much more powerful folder technologies are now feasible. To address this new challenge and exploit this new opportunity, this paper proposes a new and very powerful dynamic folder mechanism, termed **Hubble**. Hubble fully exploits the rich data model and semantic information embedded in the XML documents. It can be applied to build folder hierarchies dynamically and to categorize XML collections precisely.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

2. Hubble Dynamic Folder System

2.1 Dynamic Folder Model

In Hubble, there are two types of folders: **design-time folders** and **runtime folders**. A design-time folder hierarchy is a tree of user defined folder criteria. A design-time folder *df* is characterized by a pair (*dn*, *dq*):

- *dn* is the name of the design-time folder.
- *dq* is the definition of the design-time folder, which is specified in XQuery.

Two functions are supported on a design-time folder *df*:

- parentDf(*df*) returns the parent design-time folder of *df.*
- childDfs(*df*) returns the set of child design-time folders of *df.*

A design-time folder hierarchy represents a sketch of how a user wants to organize a collection of XML documents so that it can be efficiently searched and viewed.

After a design-time folder hierarchy is created, a user binds it to a collection of XML documents. While browsing, runtime folders are automatically created and a runtime folder hierarchy is automatically formed, according to the design-time folder definitions as well as the content of the XML documents. Similar to a conventional folder, a runtime folder contains desired XML documents in addition to child runtime folders. A runtime folder *rf* is also characterized by a pair (*df*, *rv*):

- *df* is the design-time folder that the runtime folder corresponds to.
- *rv* is the runtime value of *rf* that is defined in *df* or dynamically generated by applying *df* to the documents.

Four functions are supported on a runtime folder *rf*:

- parentRf(*rf*) returns the parent runtime folder of *rf.*
- childRfs(*rf*) returns the set of child runtime folders of *rf.*
- childDocs(*rf*) returns the set of documents contained in *rf.*
- inRfs(*doc*) returns the set of runtime folders that contain the document *doc*

Here is how childDocs(*rf*) is recursively determined, where *rf* is of a pair (*df*, *rv*):

1. Assume:
   a. *dq* is the query definition of *df.*
   b. *prf* is the result of parentRf(*rf*).
   c. *docs* is the result of childDocs(*prf*).
2. Execute *dq* on *docs*. If *rv* is a member or subset of the result of *dq*, the document is in the result of childDocs(*rf*). Otherwise it is not.

In this design, the documents in a runtime folder are a subset of the documents in its parent runtime folder.

The following describes childRfs(*rf*), where *rf* is of a pair (*df*, *rv*):

1. Assume *docs* is the result of childDocs(*rf*).
2. For each *df'* with (*dn'*, *dq'*) in childDfs(*df*).

3. Execute *dq'* on *docs*, which results in a sequence of values *vs'*. Each *df'* with a distinct value *rv'* from *vs'* forms a child runtime folder of *rf*.

2.2 Variable Binding Mechanism

The hierarchical nature of XML data model makes it easy to group related information. For example, when there is more than one vehicle, the make and the model of a vehicle can be grouped in a Vehicle element. In Hubble, we use a **variable binding mechanism** to exploit the XML grouping feature. In the definition of a design-time folder *df*, a user can create variable bindings in addition to the query definition. A variable binding is of a pair (*$var*, *vq*):

- *var* is the name of the variable.
- *vq* is an XQuery query and the variable is bound to each value in the result sequence (same as the semantics of the *For* clause in XQuery).

The variables are visible to the definition of *df* and its descendant design-time folders. With this variable binding mechanism, a design-time folder can reference the same XML element that has been referenced in a different design-time folder definition.

Here is an example which demonstrates both design-time and runtime folders. Figure 1 shows the design-time folder hierarchy which categorizes the claim documents in the Claims collection based on the make of vehicles and the damage types.

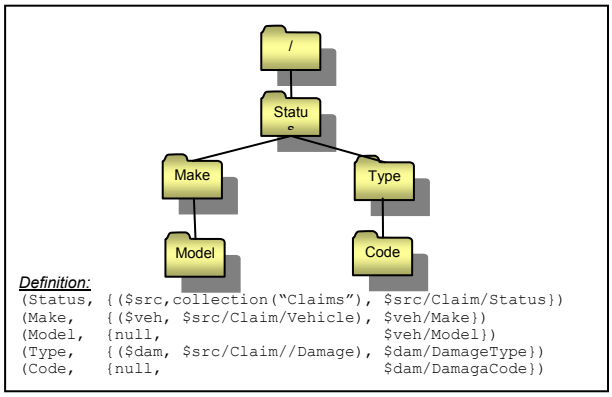

Figure 1 A design-time folder hierarchy example

Assume the Claims collection has the following document.

```
<Claim>
   <Status>In-process</Status>
   <Vehicle> <VID>J1100110011</VID>
      <Make>Honda</Make> <Model>Accord</Model>
   </Vehicle>
   <Vehicle> <VID>V1123144009</VID>
      <Make>Ford</Make> <Model>Focus</Model>
   </Vehicle>
   <Adjustment> <Damage>
        <DamageType>NonSevere</DamageType>
        <DamageCode>2</DamageCode>
   </Damage> </Adjustment>
</Claim>
```

The runtime folder hierarchy shown in Figure 2 is automatically generated at run time when it is accessed.

By binding `$veh` to a Vehicle element, `$veh/Make` and `$veh/Model` refer to the make and the model of the same vehicle. Therefore the proper relationship among the folders is maintained.

Here is the query which locates the documents under the */Status.In-process/Make.Ford/Model.Focus* folder:

```
for $src in collection("Claims"),
    $veh in $src/Claim/Vehicle
where $src/Claim/Status = "In-process"
     and $veh/Make = "Ford" and $veh/Model = "Focus"
        return $src
```

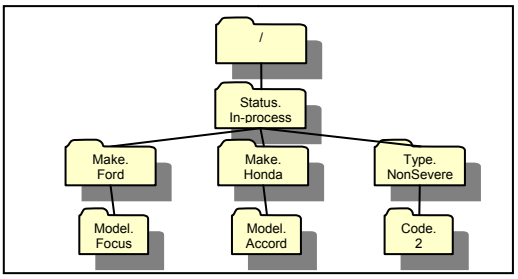

Figure 2 A runtime folder hierarchy example

3. Advanced Operations in Hubble

Two advanced operations are supported in Hubble. The first one enables users to navigate or browse runtime folder hierarchy along multiple folder paths. The second one allows folder operations to be applied to more than one document collection.

3.1 Multi-Path Navigation

Conventional navigation on a folder hierarchy allows users to follow a single path of folders and examine documents one folder at a time. However, users may be interested in the common subset of documents along multiple paths. We call this type of navigation ***multi-path navigation***. During multi-path navigation, users can define set operations on the multiple folders. There are two sensible semantics for a set operation on multiple runtime folders: the instance-based semantics, and the definition-based semantics.

3.2 Advanced Operations on Multi-Collections

In the design so far, the runtime folders in which a document is contained are entirely determined by the content in the document itself. However, other documents may hold related information that will help in categorization. Furthermore, users may want to browse into related documents which are themselves well categorized. To satisfy these requirements, our design also supports folder definitions to not only reference XML documents in other collections, but also traverse to documents in different collections.

4. Conclusion

This paper describes a flexible and powerful dynamic folder technology, termed *Hubble*, which peeks deep into the detail of XML documents to precisely categorize or group the documents. Besides supporting basic folder operations, Hubble also provides a set of advanced functionalities for organizing and categorizing documents such as multi-path navigation and folder traversal across multiple document collections. Our preliminary experiment shows that Hubble is both efficient and scalable. With the advanced functionality and efficient operations, Hubble is a viable technology for automatically categorizing XML document collections or web pages as well as dynamically building folder/directory hierarchies for XML documents.

5. References

[1] Biblioscape. http://www.biblioscape.com/

[2] J. Eder, A. Krumpholz, A. Biliris, E. Panagos. Self-maintained Folder Hierarchies as Document Repositories. *Int'l Conference on Digital Libraries: Research and Practice*, Kyoto, Japan, November 2000.

[3] Lotus Notes, http://www.lotus.com/notes

[4] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon. XQuery http://www.w3.org/TR/xquery/