Hubble: An Advanced Dynamic Folder Technology for XML

Ning Li   Joshua Hui   Hui-I Hsiao   Kevin Beyer
IBM Almaden Research Center
California, USA
Background

- Folder/directory hierarchy
  - People are familiar and comfortable with it

- Amount of information grows faster than people can manage effectively
  - Challenge: organizing large document collections such that information can be found easily and quickly

- XML and XPath/XQuery
  - Large amount of XML documents and data collections generated everyday
  - XML documents are self describing
  - New challenge and new opportunity
Hubble Objective

- Fully exploit data model and semantics of XML
- Build and manage folders dynamically and efficiently
- Precise categorization and powerful features
Example Folder Hierarchy
Hubble Core

- **Design-time folder** $df = (dn, dq)$
  - $dn$: name of the design-time folder
  - $dq$: definition of the design-time folder specified in XQuery

  - **Functions supported on $df$**
    - $parentDf(df)$ returns the parent design-time folder of $df$
    - $childDfs(df)$ returns the set of child design-time folders of $df$

- **Runtime folder** $rf = (df, rv)$
  - $df$: design-time folder that $rf$ corresponds to
  - $rv$: runtime value either defined in $df$ or dynamically generated

  - **Functions supported on $rf$**
    - $parentRf(rf)$ 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 $doc$
Design-time Folder Hierarchy

- Status: return /Claim/Status
- Make: return /Claim/Vehicle/Make
- Damage: return /Claim/Damage/DamageType
Runtime Folder Generation

<Claim>
  <Status>Completed</Status>
  <Vehicle>
    <Make>Honda</Make>
    <Model>Accord</Model>
  </Vehicle>
  <Vehicle>
    <Make>Ford</Make>
    <Model>Focus</Model>
  </Vehicle>
  <Damage>
    <DamageType>Severe</DamageType>
    ...
  </Damage>
  ...
</Claim>

Runtime Folder Management
- Automatic creation/deletion
Runtime Folder Maintenance

- Status
  - Completed
- Make
  - Honda
- Damage
  - Severe

Runtime Folder Management
- Automatic creation/deletion

XML Structure:

```xml
<Claim>
  <Status>Completed</Status>
  <Vehicle>
    <Make>Honda</Make>
    <Model>Accord</Model>
  </Vehicle>
  <Damage>
    <DamageType>Severe</DamageType>
  </Damage>
</Claim>
```
Reverse Navigation

Runtime Folder Management
- Automatic creation/deletion
- Auto-identification of runtime folders that contain an item
Extend the Example

```
Status: return /Claim/Status

Make: return /Claim/Vehicle/Make

Model: return /Claim/Vehicle/Model

Damage: return /Claim/Damage/DamageType
```
Extend the Example

```
<Claim>
  <Status>Completed</Status>
  <Vehicle>
    <Make>Honda</Make>
    <Model>Accord</Model>
  </Vehicle>
  <Vehicle>
    <Make>Ford</Make>
    <Model>Focus</Model>
  </Vehicle>
  <Damage>
    <DamageType>Severe</DamageType>
  </Damage>
</Claim>
```
Variable Binding Mechanism

Extend definition of a design-time folder

- **Variable bindings**
  - Name of the variable
  - Definition of the variable specified in XQuery

- **Query definition**
  - Use variables defined in this folder or any ancestor folder
Example with Variable Binding

```
<Claim>
  <Status>Completed</Status>
  <Vehicle>
    <Make>Honda</Make>
    <Model>Accord</Model>
  </Vehicle>
  <Vehicle>
    <Make>Ford</Make>
    <Model>Focus</Model>
  </Vehicle>
  <Damage>
    <DamageType>Severe</DamageType>
  </Damage>
...  
</Claim>
```
Advanced Operation

- Multi-path navigation
  - Operations
    - Intersection
    - Union
    - Difference
  - Semantics
    - Instance-based semantics – set operation on the documents contained in the runtime folders
    - Definition-based semantics – set operation on the query definitions
Multi-path Navigation Example

Add design-time folder Year
- Query: $v/Year
Instance-based Semantics

for $doc$ in context(),
    $v1$ in $doc$/Claim/Vehicle,
where $doc$/Claim/Status = “completed” and
    $v1$/Make = “Honda” and
    $v1$/Model = “Civic”
return $doc$

intersect
for $doc$ in context(),
    $v2$ in $doc$/Claim/Vehicle
where $doc$/Claim/Status = “completed” and
    $v2$/Make = “Honda” and
    $v2$/Year = 2001
return $doc$

for $doc$ in context(),
    $v1$ in $doc$/Claim/Vehicle,
    $v2$ in $doc$/Claim/Vehicle
where $doc$/Claim/Status = “completed” and
    ($v1$/Make = “Honda” and
     ($v1$/Model = “Civic”) and
     ($v2$/Make = “Honda” and
      $v2$/Year = 2001))
return $doc$
Definition-based Semantics

for $doc$ in context(), $v$ in $doc/Claim/Vehicle$ where $doc/Claim/Status = "completed"$ and $v/Make = "Honda"$ and ($v/Model = "Civic"$ and $v/Year = 2001$) return $doc$

$<Claim>
  $<Status>Completed</Status>
  $<Vehicle>
    $<Make>Honda</Make>
    $<Model>Accord</Model>
    $<Year>2001</Year>
  $</Vehicle>
  $<Vehicle>
    $<Make>Honda</Make>
    $<Model>Civic</Model>
    $<Year>2003</Year>
  $</Vehicle>
...$</Claim>
Multi-collection Operation –
Folder definition referencing documents in other collection

Photo Collection
<Photo>
  <Who>
    <Name>Linda</Name>
    <Name>Sandy</Name>
  </Who>
  <Where>
    <Park>Yosemite</Park>
    <State>CA</State>
  </Where>
  ...
</Photo>

Contact Collection
<Contact>
  <Name>Linda</Name>
  <Relation>Work</Relation>
  <Affiliation>IBM</Affiliation>
  ...
</Contact>
Multi-collection Operation –
Folder traversal to folders/documents in other collections
Summary

- Hubble
  - Advanced *dynamic folder technology*
  - Peek deeply into the detail of XML documents
  - Effectively and *precisely categorize* XML document collections
  - Provide advanced functions
    - Variable binding technology
    - Multi-path navigation
    - Multi-collection search and navigation
  - Experiments show efficiency and scalability
Questions ?
Hubble in Outer Space