XML/DBC: A Standard API for Access to XML Repositories and Mediators

Invited Panel, 2nd Workshop on Data Integration over the Web (DIWEB'02)

Anthony Tomasic
tomasic@e-xmlmedia.com

Introduction

The XML marketplace recently has witnessed a rapid growth in the number of XML repositories and mediators [W98] based on XQuery and XPath. In addition to a specification for the query language, a uniform API for the access to repositories, mediators, and wrappers is needed to insure interoperability of these products from different vendors. This paper proposes XML/DBC [XML/DBC] as an API for access to XQuery and XPath based XML repositories, mediators and wrappers. The API comes in both a Java class library form and in a WSDL Web Services form. XML/DBC is intended to cover multiple different repository and mediator technologies [XML:DB]. See http://www.e-xmlmedia.com/xmldbc.html for complete details.

Repository Creation

The XML/DBC API provides a repository abstraction as a collection of XML collections. Thus a repository can create, delete, etc. collections. The following call sequence creates a new collection in the repository using a default configuration (cf. Metadata below).

```
XMLRepository repository = new Repository(…);
XMLCollection collection = repository.createCollection("auction",
    "An Auction Collection", new XMLConfiguration());
```

Updates

Once a collection exists, documents are inserted into it. The simplest method provides and identifier and the XML document as a JAXP org.xml.sax.InputSource object.

```
collection.insertDocument("auction0", new InputSource("auction.xml"));
```

To access this document, the JAXP interface uses the XML Schema described in the `auction.xml` document to automatically construct a SAX parser. The result of this method is the insertion of a document into the collection and thus into the repository.

Queries

A straightforward interface, derived from the JDBC standard, provides query access. The data model consists of a single root note with a child for each collection. Each child collection is the sequence of documents in that collection. This one level of indirection permits queries to easily span multiple repositories.

In this example a repository connection is acquired and a statement generated that accepts XPath queries. The query is executed on the statement and a boolean result is returned. Since the result set generally is not an XML document, support is provided for creation of a legal document from the node sequence that is the result of an XPath expression. The XML document is then converted into a DOM document. (SAX support is also available.) The result of execution is the `auction.xml` document.

```
XMLConnection xc = repository.getXMLConnection();
XMLStatement xs = xc.createStatement(XPATH_QUERY_TYPE);
boolean result = xs.execute("/auction");
if (result) {
    XMLResultSet xrs = xs.getResultSet();
    XMLDocument xd = xrs.getAsXMLDocument("namespace",
        "localname", "qualifiedname", "root");
    Document d = xd.getAsDOM();
}
```

Mediators

Mediators use the same abstractions as repositories from the XML/DBC API. A data source is interpreted to be an XML collection. In this example, the interface `XMLMediator` shares the same interface as the repository. The parameters supplied to configuration are sufficient for a wrapper to attach to the mediator. For example, in the case of a relational database the configuration arguments would include a JDBC connect string, login name and password.

```
XMLMediator mediator = new Mediator(…);
XMLConfiguration xc = new Configuration(…);
XMLCollection collection = mediator.createCollection("auction",
    "An Auction Collection", xc);
```

Subsequently, all the above examples execute in the same way with the Mediator as with the Repository. However, some collections may not accept updates. To handle this issue and others, metadata describes the properties of collections.

Metadata

Since collections correspond to data sources in mediators and wrappers are used to implement data sources, metadata implicitly describes the properties of a wrapper and its data source. A predefined set of metadata properties permits the data source to easily declare support for various common cases, such as support for XQueries, XPaths, updates, multiple collections, etc. A `getSchemaNamespaces` method returns a list of name spaces, and for each name space, a `getSchema` method returns the schema for that name space. A `getCapabilities` method returns a description of the query capabilities [TRV98] of a data source. The exact definition is still open but our goal is to provide a capability language so that JCA, WSDL, IMS, HTTP and other standards can be easily integrated as data sources. Finally, note that mediators themselves have metadata and thus are also wrappers for other mediators.

References

[TRV98] Anthony Tomasic, Louiqa Raschid, Patrick Validuriez, "Scaling Access to Heterogeneous Databases with DISCO," in *IEEE Transactions on Knowledge and Data Engineering*, 10(5), 1998.

[W98] Gio Wiederhold, "Mediators in the Architecture of Future Information Systems," IEEE Computer 25(3): 38-49, 1992.

[XML/DBC] Arnaud Witschger, Anthony Tomasic, "E-XMLMedia XML/DBC API Proposal," draft unpublished manuscript, 2002. See http://www.e-xmlmedia.com/xmldbc.html

[XML:DB] XML Database API Project. See http://www.xmldb.org/