DynamicWrappers: a rule based metadata conversion and integration system for e-Science applications

Shoaib Sufi
CCLRC – Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, UK.
E-mail: s.a.sufi@dl.ac.uk

Abstract: *A need to integrate heterogeneous metadata/relational data sources on the Grid and in e-Science Projects exists Often this is done in a time consuming and hard-coded way. What is proposed is a generic dynamic configuration driven Grid aware system which initially supports the transformation of metadata from any Relational Schema to any XML Schema. Existing Transformation systems are either manual or not fully featured enough to support anything but fixed non-recursive structures as well as not being Grid aware. A configuration based approach is not only amenable to further tool support but makes homogenisation to common standards require less man effort than bespoke solutions. A configuration based system will be a natural fit to some of the important data management requirements of many Grid/e-Science projects.*

1. Introduction

A need to integrate heterogeneous metadata sources on the Grid and in e-Science Projects exists. Often this is done in a time consuming and hard-coded way; this is not only costly but generally an uninteresting piece of work which requires a fair amount of 're-inventing the wheel' and can be error prone.

DynamicWrappers have been developed as part of the CCLRC core funded DataPortal project[1] they were formally known as XMLWrappers. DynamicWrappers are a generic dynamic configuration driven Grid aware system which initially supports the transformation of metadata from any Relational Schema to any XML Schema[2].

Existing Transformation systems are either manual[3] or not fully featured enough to support anything but fixed non-recursive structures[4] as well as not being Grid aware. A configuration based approach is not only amenable to further tool support e.g. GUI's for writing the configuration information but makes homogenisation to common standards require less man effort than bespoke solutions. Also It is to be noted that XSLT only deals with transformation from XML.

A configuration based system will thus be a natural fit to some of the important data management requirements of many Grid/e-Science projects which are aiming to federate access to (often heterogeneous) metadata/relational data via Web/Grid Services and XML mediation.

2. Architecture

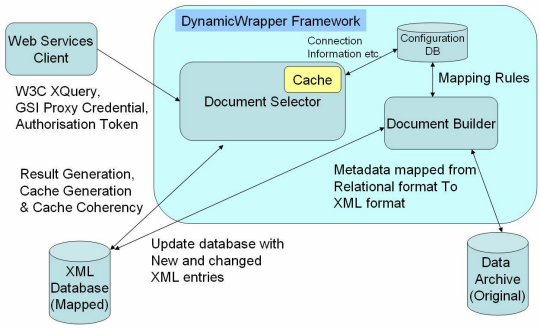

Figure 1: DynamicWrapper Architecture

Each DynamicWrapper instance consists of two parts: a DocumentSelector and a DocumentBuilder.

The DocumentBuilder general operation:

1. Connecting to the source data repository (i.e. Relational database)
2. Connecting to the XML:DB aware database repository
3. Reading the configuration information from relational configuration tables
4. Dynamically creating the SQL statements needed to map the SQL in the source database to the required XML Schema format.
5. Perform the mappings
6. Periodically detect new, updated and deleted records and update the XML

database accordingly in a cache coherent manner.

The DocumentSelector is the Grid enabled interface to the XML:DB Mapped Archive. It's mode of operation is generally as follows:

1. Connect to the XML database
2. Can be invoked via Web Services with the following arguments:
   2.1. The GSI Proxy certificate
   2.2. The DataPortal Authorisation Token (an XML-Signature signed file containing role information)
   2.3. The W3C XQuery

(note: it also supports a Query to extract the location of the XML Schema of the documents in the XML Database to allow schema discovery).

3. Proxy validation and group membership are checked
4. The confirmed group membership is checked against the permissions associated with this role. The part is still bespoke but it is hoped that a mediation layer will be developed as another part of the DataPortal Project and this could be then be configuration based. Access Control Lists are seen as being in a possible mechanism. The development of such a mechanism is not currently part of the DynamicWrapper framework.
5. Once the query is authenticated and authorised the XQuery is applied to the XML Database and the results returned via SOAP.

3. Configuration

Configuration of the DynamicWrapper is via a number of relational tables. What follows is a short overview pertaining to some of their names and functions:

Table 1: Configuration Tables

Table Name	Purpose
SQL_INFO	Holds the details needed to extract one value from one or more columns with dependencies on other mapping rules in the SQL_INFO table and ways of dealing with recursive values specified. Thus necessary nested join information is specified.
SQL_TO_XML	Contains a set of references to the SQL_INFO rows and XPath locations such that a valid instance document can be constructed
STX_SCHEMA	Contains details about the schema which the instance XML documents conform to such as the namespace and location of the schema

There are other table also holding housekeeping configuration information such a connection parameters and cache serialised objects.

This flexibility of the mapping tables allows specifying mappings from any relational schema to any XML Schema.

3.1. Configuration Example

A common SQL statement that is needed to 'walk' a relational hierarchy is the following:

```
select name from topic where
topid_id in (select topic_id from
topic_investigation where
investigation_id in(select
investigation_id from
investigation_study where study_id =
'23')) ;
```

Figure 2: Typical Select

Such SQL statement are typical when extracting relevant values to apply to a target XML instance e.g. the name of a topic linked to a study with study_id='23' in this case.

The main configuration table in the DynamicWrappers is called SQL_INFO and has the following structure:

Table 2: Detailed Structure of SQL_INFO

Column	Purpose	Key
SQL_INFO_ID	Numeric primary key column	**1**
PARENT_SQL_INFO_ID	TAB_INFO_ID of parent entry	**2**
COLUMN_NAME	Column name in the table	**3**
TABLE_NAME	Table name of the	**4**

	column	
LINK_COLUMN	Name of column that does the linking	**5**
CONCAT_VALUE	Character sequence that joins result tokens in recursive specification where TAB_INFO_ID and PARENT_TAB_INFO_ID are the same value	**6**
VALUE_ORDER	For recursive SQL is the result base->first or base->last	**7**
BASE_LINK_COLUMN	Specified the column in the table which is used to link this row to the key of other rows in recursive calls	**8**
BASE_LINK_COLUMN_VALUE	Again for recursion; specifies what the link column value is in the base case (i.e. when you stop recursion)	**9**

The follows shows the representation of the SQL in Figure 2 in the SQL_INFO table. (note columns whose values would be null are omitted).

Table 3: SQL_INFO representing a normal select

1	2	3	4	5
44	34	name	topic	topic_id
34	23	topic_id	topic_investigation	investigation_id
23	-1	investigation_id	investigation_study	study_id

The DocumentBuilder would select the appropriate SQL_INFO_ID (SII) and an end statement value when building the SQL from this configuration.

The algorithm used to construct the SQL from the information in SQL_INFO is as follows:

1. select <column_name> from <table_name> where <link_column>

2. if parent_sql_info_id is a special value (e.g. in this case -1) then the final value of link_column must be set =’some value supplied externally’ other wise we concatenate the sequence ‘IN (‘

3. do 1 and 2 until you reach the base case of PARENT_SQL_INFO_ID having a special value and then tidy up any remaining brackets.

Thus in the above example we would have:

1. (PSII=44) – select name from topic where topic_id in (
2. (PSII=34) – select topic_id from topic_investigation where investigation_id in (
3. (PSII=23) – select investigation_id from investigation_study where study_id = ‘<some value>’
4.)) ;

Thus mirroring the above SQL statement in Figure 2.

The configuration scheme also deals with nested values of arbitrary depth. The following being the specification for a recursive approach to building the value which is mapped to an XML Schema instance:

Table 4: SQL_INFO representing a recursive select

1	2	3	4	5	6	7	8	9
45	45	topic_name	topic	topic_id	/	first	parent_id	-1

Basically this specifies building up the value that is later mapped to an XML instance element or attribute (or any other target) recursively. The algorithm expressed in pseudo code for this being (note: `val` is initially supplied):

```
do {

select <column_name>,
<base_link_column> from <table_name>
where <link_column> = val ;

/* .at_start – where base->first,
.at_end where base->last */

result.at_start(<concat_value>+<colu
mn_name>) ;

val = <base_link_column> ;

}while (val !=
<base_link_column_value>) ;

return result ;
```

Figure 3: Building recursive values

The complete configuration schema specifies mappings from a relational schema to an XML instance document. XPath is used to specify actual locations that values are mapped to in XML instance documents and other necessary aspects to actual create instance document:

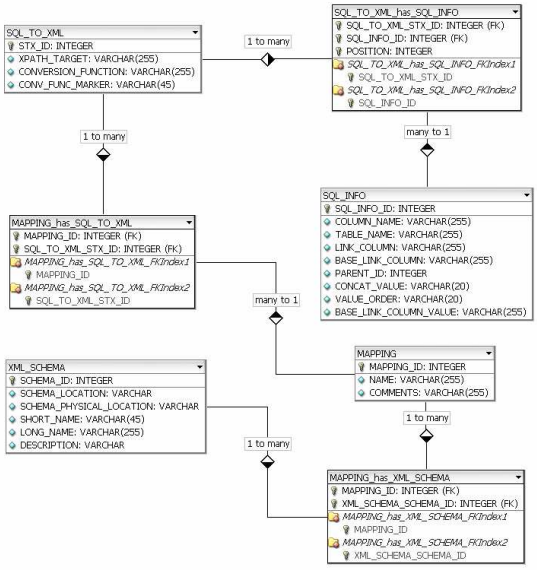

Figure 4: Detailed Dynamic Wrapper Configuration Schema

4. Future

The Dynamic Wrappers will be used in the Internal Facilities of CCLRC as part of the DataPortal project with possible application to the e-Minerals project [5]. OGSA-DAI could be used as an alternative DocumentSelector but the DocumentBuilder mapping ability is not generally available functionality. DynamicWrappers are generic middleware and have the potential to be used on a variety of projects requiring metadata and relational data transformation services.

5. References

[1] The CCLRC DataPortal, http://www.allhands.org.uk/2004/proceedings/papers/161.pdf

[2] W3C XML Schema, http://www.w3.org/XML/Schema

[3] JnetDirect JSQLMapper, http://www.jnetdirect.com/products.php?op=jsqlmapper

[4] Oracle XML Data Synthesis (XDS), http://www.oracle.com/technology/tech/xml/xds/index.html

[5] Environment from the Molecular Level, http://eminerals.org/