BIRT XML Data Source Design Specification

Design Specification

Draft 3: Sep 12, 2005

Abstract

This is the design specification of the BIRT XML Data Source.

Document Revisions

Version	Author	Date	Description of Changes
Draft 1	Gary Xue	Aug 8, 2005	First draft
Draft 2	Gary Xue	Aug 22, 2005	Added examples, xpath expression restrictions
Draft 3	Gary Xue	Sep 12, 2005	UI Mockup; minor updates to runtime specs

Contents

1. Introduction 3

2. Features and Requirements 3

2.1 Features Committed to in BIRT 2.0 3

2.1.1 Any Source of XML Data 3

2.1.2 Versatile XML to Relational Mapping 3

2.1.3 Multiple Tables in One Data Source 3

2.1.4 Parameterized Queries 4

2.1.5 Visual Tool to Create Relational Mapping 4

2.1.6 Support Large XML Data 4

2.2 Tentative Features 4

2.2.1 Runtime Validation 4

2.2.2 XML Includes and Imports 4

2.3 Features for Future Consideration 4

2.3.1 Emitted XML Data 4

2.3.2 Hierarchical Data 4

2.3.3 Supporting Filtering and Sorting 4

2.3.4 Table Joins 5

2.3.5 Multiple Stream in One Data Source 5

3. XML to Relational Mapping 5

3.1 Table and Columns 5

3.2 XPath Expressions 5

3.3 Table Row Mapping 5

3.4 Column Mapping 6

3.5 Parameterized Mapping 6

3.6 Examples 6

3.6.1 Table and Column Mapping Examples 7

3.6.2 Data Set Parameter Example 8

3.7 Restrictions on XPath Syntax 8

3.8 Dealing with Nested XML Data 9

3.8.1 Mapping of Nested XML Data 9

3.8.2 Automatic Mapping of Nested XML Data 10

4. XML Data Source Designer 10

4.1 Creating a XML Data Source 10

- *Source of Data 10*
- *XML Schema 10*

4.1.1 UI Mockup 10

4.2 Creating a XML Data Set 11

4.2.1 Selecting Sample Source Data 11

4.2.2 Define Relational Mapping 12

5. ODA Extension Design 16

1. Introduction

The prevalence of XML as the standard data format for data exchange among application, and increasingly as a data storage format, has made XML an ever-important source of data for reporting. The XML Data Source project aims to provide BIRT with out-of-the-box access to XML data. We have defined several goals for this project, scheduled to be completed as part of BIRT 2.0:

Enables BIRT to report against most common types of XML data, with no programming required on the part of the report developer, and minimum amount of configuration.

Deliver another architecturally sound open source ODA driver to demonstrate the fully capability of the ODA framework, and to serve as a template for future community development of specialized data access drivers.

The XML Data Source is built on top of the Open Data Access (ODA) framework. It is delivered in two parts: an ODA runtime driver (the “XML Data Source Runtime Driver” or the “Driver”) to enable runtime access to XML data as a report data source, and a design-time BIRT UI plug-in (the “XML Data Source Designer”, or “Designer”) for visual building of XML data source metadata at design time.

The XML Data Source project seeks to leverage as much as possible established standards and existing open source implementations.

2. Features and Requirements

2.1 Features Committed to in BIRT 2.0

2.1.1 Any Source of XML Data

The Driver should be able to access XML data from a URL, which can be provided by any underlying source.

For ease of use, flat-file source of XML data is presented in XML Data Source Designer as a separate source of data, even though it can also be expressed as a URL using the file protocol.

2.1.2 Versatile XML to Relational Mapping

The XML Data Source makes very little assumption about the structure of the XML data. By using XPath expressions to create XML to table/column mapping, most common types of XML data can be mapped to tables and columns without the need for additional transform or programming. (See Section 3).

2.1.3 Multiple Tables in One Data Source

The XML Data Source views each XML data stream as a collection of row sets, each row set having a fixed set of fields. A row set is alternately referred to as a *Table* having a fixed set of *Columns.* Multiple tables can be defined in one data source.

2.1.4 Parameterized Queries

The XPath expressions used to flatten the XML can contain parameters whose values are supplied at runtime. This can be used to take full advantage of XPath search capabilities.

2.1.5 Visual Tool to Create Relational Mapping

The XML Data Source Designer provides a visual tool to help create a mapping from XML to relational data. To enable the visual tool, the user must provide either a XML Schema file, or a sample XML data file, or both. If an XML data file is provided, the visual tool will also provide data preview.

2.1.6 Support Large XML Data

The Driver must be able to support large XML data streams, whose content may not be able to fit in memory at one time.

2.2 Tentative Features

2.2.1 Runtime Validation

If an XML Schema file is provided to the Designer, the same schema file can optionally be used to validate the XML data at runtime. XML data that fails the validation will cause an error to be generated at runtime.

2.2.2 XML Includes and Imports

Many common XML formats (e.g., XSD, WSDL) define *include* or *import* elements to incorporate the content of another XML source. The XML Data Source should be able to handle some common

2.3 Features for Future Consideration

2.3.1 Emitted XML Data

Allow access to XML data emitted by Java objects.

2.3.2 Hierarchical Data

Allow row-level data to be hierarchical (non-flattened). Enable BIRT Script access to such structured data as JavaScript objects. This requires enhancements to ODA and BIRT.

2.3.3 Supporting Filtering and Sorting

Support filtering of data (beyond what XPath is capable of doing) and sorting in the Driver.

[Note: This duplicates what BIRT data engine is capable of doing, and adds very little value to the product, since the driver most likely cannot sort/filter any more efficiently than the Data Engine can.]

2.3.4 Table Joins

This calls for multiple tables to be used in one data set, which allows the driver to join them in one query through the use of user-defined join conditions.

[Note: The more appropriate place to implement this feature is in the BIRT data engine, which should be able to join data from multiple data sets. It's a more generic solution that a join done in the XML driver]

2.3.5 Multiple Stream in One Data Source

Certain application split large XML data into multiple files or streams. The XML Data Source should be able to merge the content of multiple streams and view it as a single XML data stream.

3. XML to Relational Mapping

In order to for BIRT, which at present only deals with relational data sources, to access XML data, the XML Driver must flatten XML data and present it in a relational model, i.e., as tables and columns. XML to relational mapping is the most important aspect of the driver design. The user defines rules to map XML data to relational entities with the help of the XML Data Source Designer.

3.1 Table and Columns

An XML data source, i.e., an XML data stream, is mapped to one or more *tables*. Each table contains a collection of rows with identical column structure. Each column has a name and an ODA data type.

The XML Data Source assumes that *each data row is mapped to one XML element*. Columns in the data rows are typically mapped to either child elements or attributes within the data row element.

3.2 XPath Expressions

The XML Data Source uses XPath expressions to define mappings of tables and columns. XPath is a W3C standard for locating XML data. Its powerful search capability, and the wealth of open source standards-conforming implementation, makes XPath an ideal choice for defining XML to relational mapping.

3.3 Table Row Mapping

A table is a collection of rows with identical column structure. Mapping from XML data to rows in a table is defined by a single XPath expression which must evaluate to a set of XML elements. Each element in the result set of the XPath evaluation maps to a row in the table. The XPath expression that defines the element to row mapping must be an absolute location path, which starts with "/". If the table mapping XPath expression is not an absolute path, or if it evaluates to anything other than a collection of element nodes, a runtime error will result.

Filtering of data rows can be achieved though the use of pattern matching and predicates in XPath.

The Driver returns rows in the same order as their mapped XML elements appear in the XML stream.

3.4 Column Mapping

A table's definition also includes a set of columns. A column's definition comprises of three parts:

(1) Column name. This is a string value provided at design time.

(2) Column data expression. This is defined by a single XPath expression. The runtime Driver evaluates this expression using the XML element which maps to a data row to determine the value of the corresponding column in that row. It is typically a relative XPath location that identifies an attribute or a child node of a row element. However it is also possible for this expression to refer to attributes or elements along the parent axis. Section 3.8 gives an example of such usage.

If the column data XPath expression evaluates to an attribute node, the value of the attribute is used as the column value. If the expression evaluates to an element node, the text value of the element is used as the column value. If the expression evaluates to a scalar value, that value is used as the column value. If the expression evaluates to any other result (such as a collection of elements), a runtime error will result.

(3) Data type. This must be one of the data types supported by ODA. It is provided at design time.

3.5 Parameterized Mapping

XPath expressions that define table or column mapping can contain parameters, whose values are supplied at runtime. This allows the mapping to be dynamic. This can be used, for example, to pass down runtime filtering conditions to the driver.

A parameter is identified by pattern **{?*param_name*?}** in the XPath expression. The Driver searches the mapping expressions for pattern strings, and presents them as data set query parameters. At query execution time, parameter patterns are replaced with actual parameter values before the XPath expressions are evaluated.

3.6 Examples

The following sample XML text is used to illustrate the examples in this section.

```
<?xml version="1.0"?>
<library>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author name=”Giada De Laurentiis” country=”it”/>
  <year>2005</year>
</book>

<book category="CHILDREN">
  <title lang=”en”>Harry Potter</title>
  <author name=”J K. Rowling” country=”uk” />
  <year>2005</year>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author name=”James McGovern” country=”us” />
  <author name=”Per Bothner” country=”us” />
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author name=”Erik T. Ray” country=”us” />
  <year>2003</year>
</book>

<audio format=”CD” category=”MUSIC”>
<title lang=”en”>Feels Like Home</title>
  <artist name=”Norah Jones” country=”us” />
  <year>2005</year>
</audio>

</library>
```

3.6.1 Table and Column Mapping Examples

Example 1:

Table XPath		/library/book	
Column	**Name**	**XPath**	**Data Type**
1	Category	@category	String
2	Title	title[@lang=’en’]	String
3	Year	year	Integer
4	Author_1	author[1]/@name	String
5	Author_2	author[2]/@name	String

In this example, the table XPath expression “/library/book” evaluates to a collection of all 4 *book* nodes under the XML root element. Each of these elements is mapped to a row in the resulting table. The 5 column data expressions are each evaluated against each *book* node to calculate the value of the respective column. Take the first *book* node as an example:

```
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author name=”Giada De Laurentiis” country=”it”/>
  <year>2005</year>
</book>
```

Expression “@category” evaluates to attribute node

category="COOKING"

The value of this attribute, “COOKING”, becomes the value of column *Category* in the first data row.

Similarly, expression “title[@lang=’en’]” evaluates to this child node of the first *book* node:

```
  <title lang="en">Everyday Italian</title>
```

The text value of this node, “Everyday Italian”, becomes the value of column *Title.*

This definition maps the source XML to the following table named “Books”.

Category	**Title**	**Year**	**Author_1**	**Author_2**
COOKING	Everyday Italian	2005	Giada De Laurentiis	NULL
CHILDREN	Harry Potter	2005	J K. Rowling	NULL
WEB	XQuery Kick Start	NULL	James McGovern	Per Bothner
WEB	Learning XML	2003	Erik T. Ray	NULL

Example 2:

Both books and audio collections are mapped to one table

Table XPath		/library/*	
Column	**Name**	**XPath**	**Data Type**
1	Title	title[@lang=”en”]	String
2	Year	year	Integer

Example 3:

Using XPath predicates for filtering, one can define table mapping XPath expression to be:

/library/book[@category=”WEB”]

The resulting table will only contain books in the WEB category.

3.6.2 Data Set Parameter Example

If the user defines the table mapping XPath expression to be:

/library/book[@year= **{?publication_year?}**]

The text in bold declares data set parameter named “publication_year” with String type. At runtime, the Driver replaces the parameter string with the actual parameter binding before evaluating the XPath expression. For example, if a parameter value of “2005” is provided at runtime, the data set’s result will only contain books having 2005 as the value of the year element.

3.7 Restrictions on XPath Syntax

All known open source XPath implementations are based on DOM, which means that the available memory size limits the size of the XML data that can be handled. In order to be able to efficiently handle large XML source, the Driver must use a SAX-based solution to evaluate XPath expressions used to define table and column mapping.

The XML Driver in BIRT 2.0 places the following restrictions on XPath expressions that can be used to define table/column mapping:

1. The expression can only locate elements along the child, descendant or parent axes. Only the abbreviated syntaxes for these axis locations are recognized. (E.g.,

"/library" ,"//book" and "../@name" are OK; "/child:library" or "/descendant:book" are not).

2. Only the following predicates are recognized:
   - A single position predicate in the abbreviated form, e.g., "author[2]".
   - A single equality condition based on an attribute value. E.g., "title[@lang='en']".
3. XPath functions are not supported

3.8 Dealing with Nested XML Data

This section describes the mapping and handling of a common type of XML data where multiple relational entities and their 1-to-many relationships are expressed in one nested tree structure. The sample XML segment below is one such example: it describes two entities, order and order item, and defines a 1-to-many mapping between them.

```
<Orders>
<Order id="1">
         <OrderDate>2005-8-10</OrderDate>
         <OrderItem>
               <Name>Stumps</Name>
               <Quantity>3</Quantity>
         </OrderItem>
</Order>
<Order id="2">
         <OrderDate>2005-8-12</OrderDate>
         <OrderItem>
               <Name>Stumps</Name>
               <Quantity>3</Quantity>
         </OrderItem>
         <OrderItem>
               <Name>Gloves</Name>
               <Quantity>1</Quantity>
         </OrderItem>
</Order>
</Orders>
```

3.8.1 Mapping of Nested XML Data

Many types of mapping are possible with nested XML data using XPath expressions. However given the BIRT Data Engine's limitation that it can only work with one data set per query (i.e., cross-data set joins are not possible), we recommend mapping the above data into a denormalized join table of order and order items:

Order_Id	***Order_Date***	***Item_Name***	***Quantity***
1	2005-8-10	Stumps	3
2	2005-8-12	Stumps	3
2	2005-8-12	Gloves	1

This mapping is created using the following definition:

Table XPath		/Orders/Order/OrderItem	
Column	**Name**	**XPath**	**Data Type**
1	Order_Id	../@id	Integer
2	Order_Date	../OrderDate	Date
3	Item_Name	Name	String
4	Quantity	Quantity	Integer

Note the use of the parent axis (in its abbreviated form "..") in the two highlighted columns mapping expressions. The Driver must allow this type of search along the parent axis in column mapping expressions.

3.8.2 Automatic Mapping of Nested XML Data

Because nested XML data is quite common in real-world application, and the less-than-intuitive way that they are mapped to relational data by use of XPath expression, the Designer must be able to assist in the creation of this type of mapping. It needs to automatically detect possible nested XML structures and automate the creation of XPath expressions to map the data into relational tables.

4. XML Data Source Designer

Since XML Data Source is implemented as an ODA extension, creating and using an XML data source is similar to using any other ODA data source. This section describes the workflow to create a XML data source and data set. The UI mockups offered in this section are for illustration purpose only and may differ from actual implementation.

4.1 Creating a XML Data Source

A new BIRT ODA data source type, "XML Data Source", is registered by the XML driver extension. The report designer first creates a data source of this type, and use the UI wizard to provide the following information that defines a XML data source.

- **Source of Data**

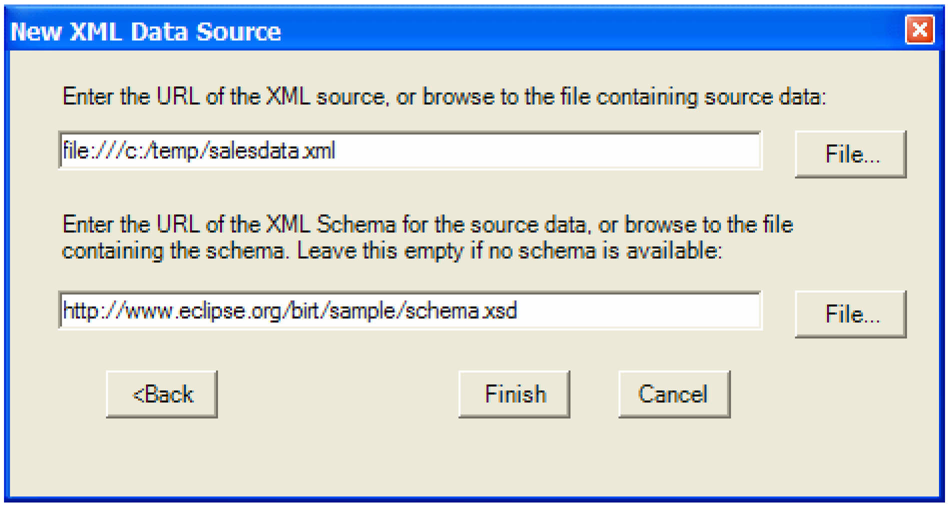
The user provides the URL that identifies the source of the XML data. The user can also give a file name, or browse to an XML data file in the local file system.

- **XML Schema**

The user provides a URL of the XML Schema (XSD) for the XML data. The Schema can also be a file in the local file system. The XML Schema is optional. Providing the schema file however is required to enable runtime validation of XML data. It also allows the data set designer to display the structure of the XML data to facilitate the creation of table/column mapping rules.

4.1.1 UI Mockup

The user sees the following dialog when creating a XML Data Source.



Enter the URL of the XML source, or browse to the file containing source data:

file:///c:/temp/salesdata.xml

Enter the URL of the XML Schema for the source data, or browse to the file containing the schema. Leave this empty if no schema is available:

http://www.eclipse.org/birt/sample/schema.xsd

Upon choosing "Finish", the designer performs the following validations and actions.

1. It attempts to connect to the URL specified for the source data. If a connection cannot be made, a warning is shown. However in invalid URL does not stop the data source from being created, since the URL may not be "live" at design time, or it may simply be an empty of stub value to be overwritten by scripts executed at runtime.
2. The Schema file URL, if not empty, is loaded. The entire content of the schema file is copied into the data source design. If the URL is not valid or if the content is not a valid XML file, an error is shown and the data source is not created.

4.2 Creating a XML Data Set

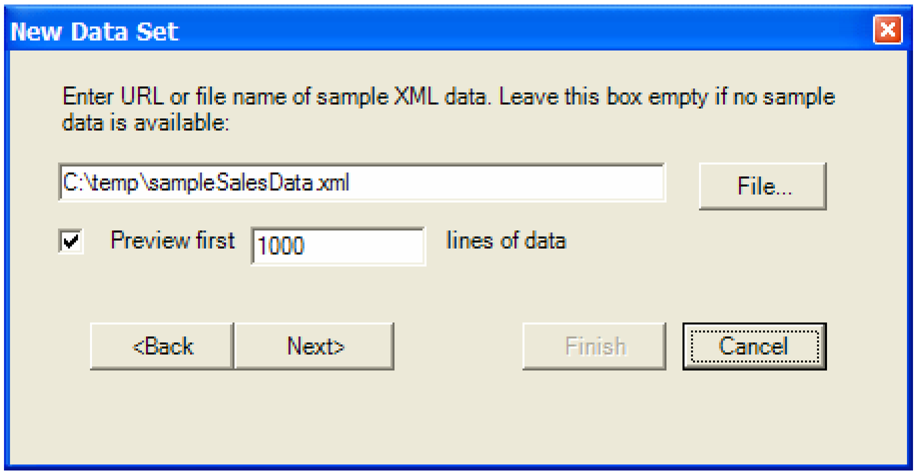
A new BIRT ODA data set type, "XML Data Set", is registered by the XML Driver extension. The New Data Set wizard in the BIRT report designer shows this data set type when a new data set is being created on an XML data source.

Each XML Data Set design defines mapping rules for one table. The XML Data Set designer is a wizard that guides the user through a series of steps to create the mapping rules.

4.2.1 Selecting Sample Source Data

To enable the data set wizard to provide help with relational mapping, the user should provide sample XML Data. The first dialog of the wizard asks the user for the URL or file name of the sample data. The URL input textbox is pre-filled with the URL defined for the data source. The user can select a different URL or file should that URL not be available at design time. If no sample data is available, the input textbox should be cleared.

If the sample data is large, the user can choose how many lines of the sample data should be used for preview purpose. By default, only first 1,000 lines of data are processed by the designer for performance reasons.


New Data Set

Enter URL or file name of sample XML data. Leave this box empty if no sample data is available:

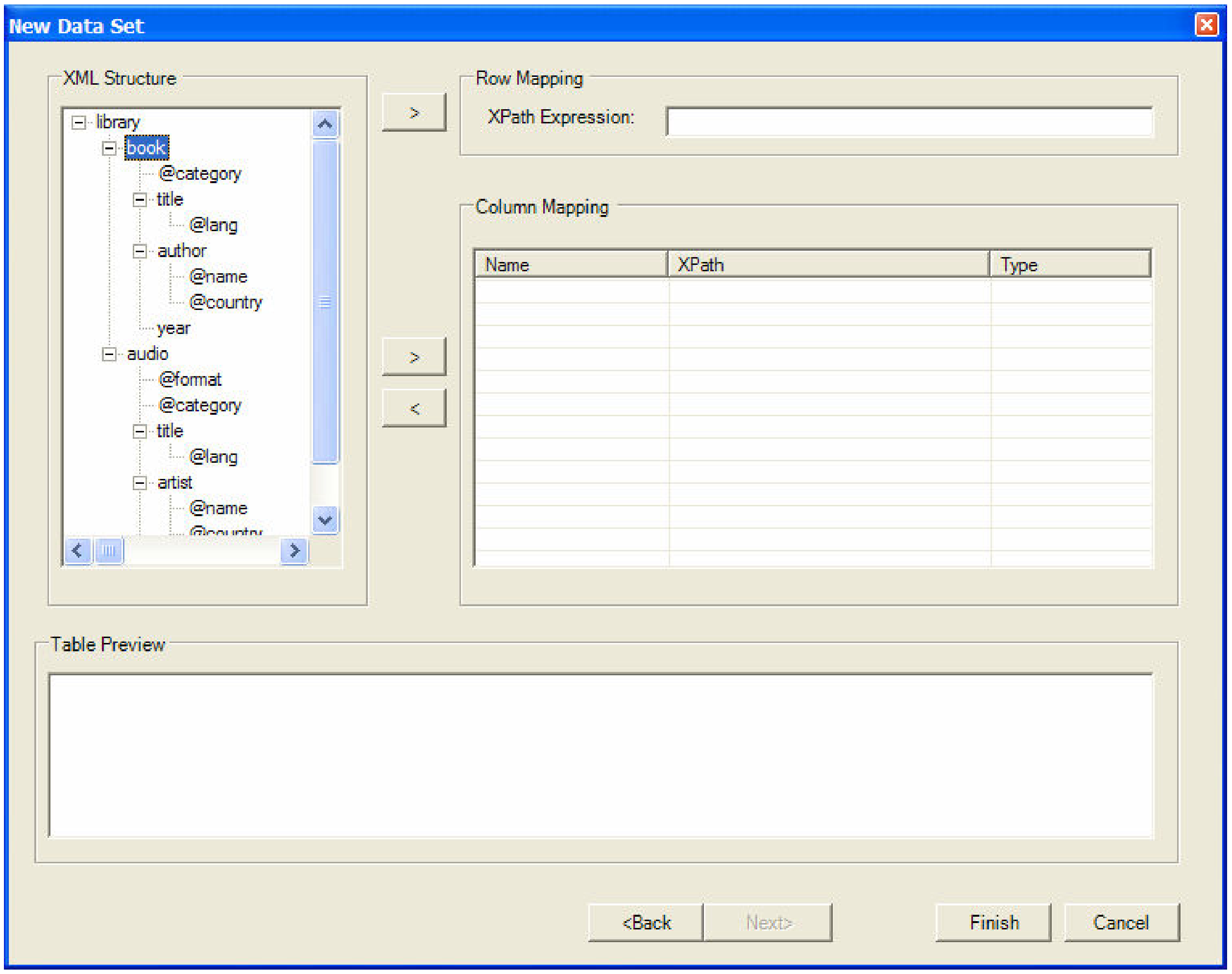
C:\temp\sampleSalesData.xml

[✓] Preview first 1000 lines of data

Upon choosing “Next>”, the designer loads the sample data. An error message is shown if the URL or file name is not valid, and control is returned to this dialog so the user can correct the problem.

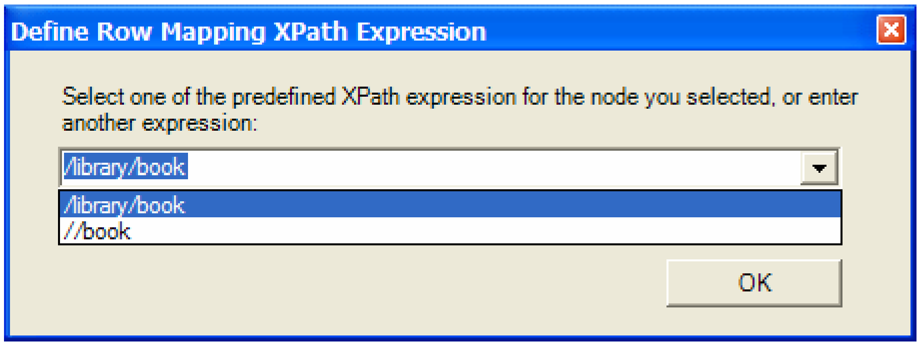
4.2.2 Define Relational Mapping

The final dialog of the wizard helps the user create mappings for the data set’s table. The dialog has a tree view that is populated with the structure of the XML data. This structure is determined by the XML Schema defined for the data source. If the schema is not available, the sample data is analyzed to extract its structural information. The structural view is empty is neither the schema nor the sample data is available. In the structural view, attributes are prefixed with the “@” sign to distinguish them from elements.



4.2.2.1 Table Mapping

The user must first define an XPath expression to create the row mapping as described in Section 3.3 . This can be done by manually entering any valid XPath expression into the Row Mapping textbox. Alternately the user may select an element in the XML Structure tree and click the upper ">" button. When this happens, a dialog pops up to help the user define the XPath expression. The following screenshot shows what happens when the user selects the "book" node and then click on the upper ">":



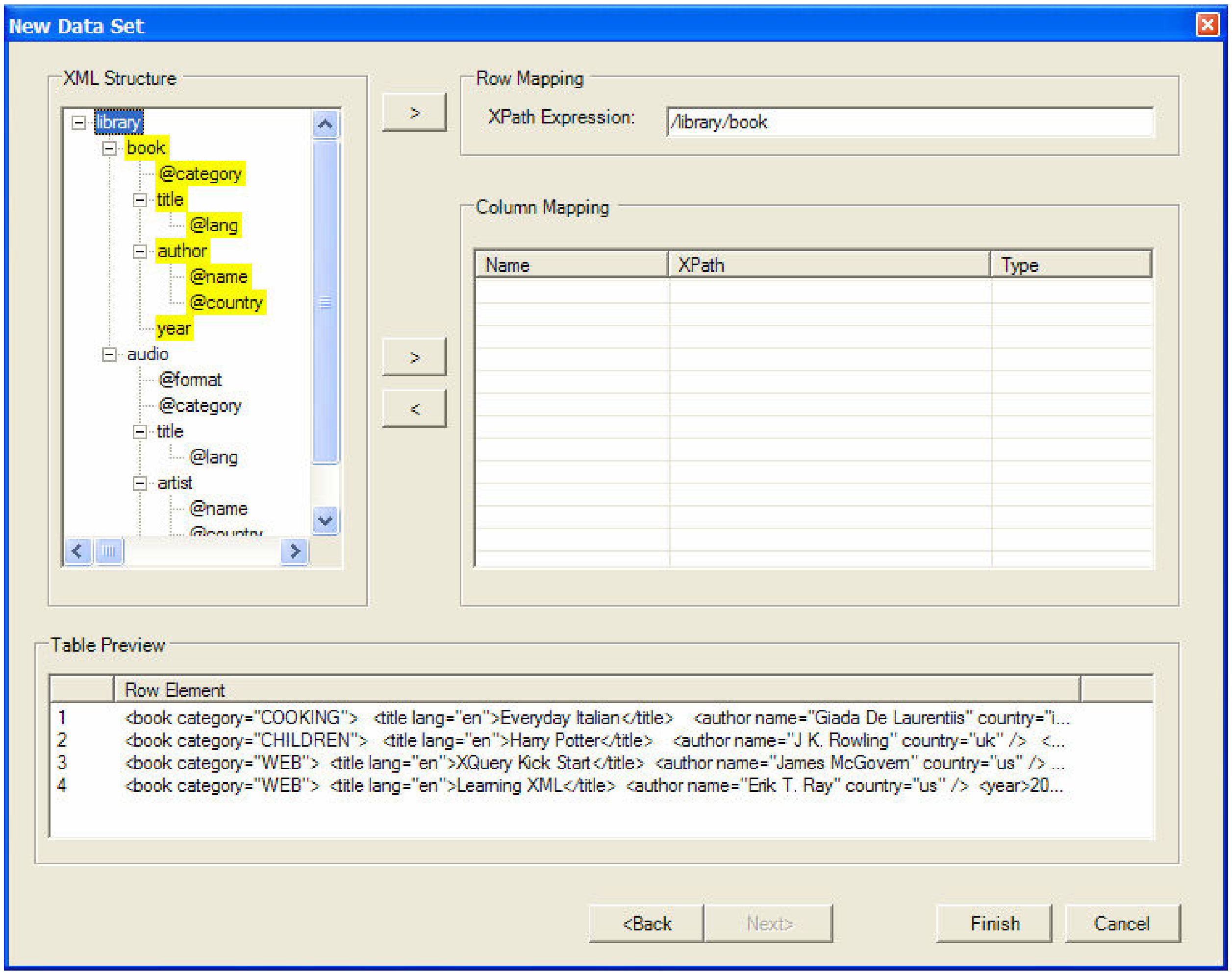
Select one of the predefined XPath expression for the node you selected, or enter another expression:

/library/book

/library/book
//book

The combo box is pre-filled with two common XPath expressions that can evaluate to the selected node. The user may also type in a different expression, or modify a pre-generated expression to apply filtering etc.

After the user defines a row mapping expression, and assuming that no column mapping is defined at this point, the Table Preview box shows a preview of XML elements that are selected by the row mapping expression. The following example shows what happens when the user defines a row mapping using expression "/library/book".



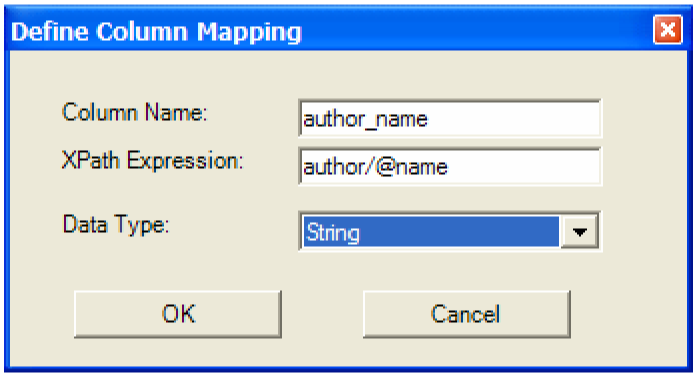
XPath Expression: /library/book

	Row Element
1	<book category="COOKING"> <title lang="en">Everyday Italian</title> <author name="Giada De Laurentiis" country="i...
2	<book category="CHILDREN"> <title lang="en">Harry Potter</title> <author name="J K. Rowling" country="uk" /> <...
3	<book category="WEB"> <title lang="en">XQuery Kick Start</title> <author name="James McGovern" country="us" /> ...
4	<book category="WEB"> <title lang="en">Learning XML</title> <author name="Erik T. Ray" country="us" /> <year>20...

Note that in the XML Structure tree, some nodes are highlighted – these nodes are elements and attributes identified by the designer as possible candidates for columns.

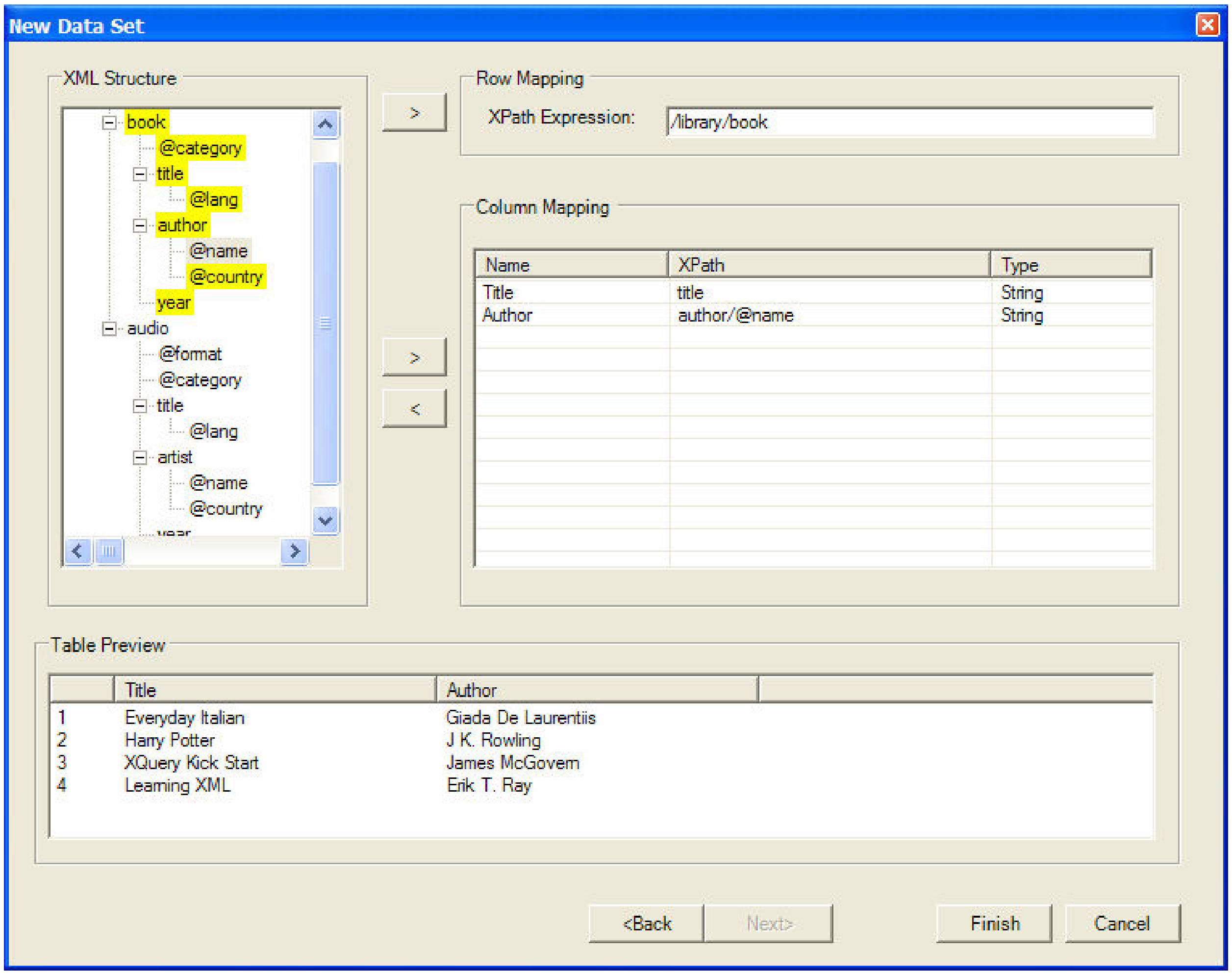
4.2.2.2 Column Mapping

Once the row mapping expression is defined, the user then proceeds to create column mappings. He can directly type in a name, XPath expression, and select a data type in the next empty row in the Column Mapping list. Or he can select a node from the XML structure view, then click in the lower ">" button to add it to the column list. The following dialog shows up when the user selects the "author/@name" node and click on the lower ">" button:



Column Name: author_name
XPath Expression: author/@name
Data Type: String

The Table Preview list is updated whenever a new column is defined, or an existing column mapping is modified. The following dialog shows what the preview list looks like after two columns have been defined:



XPath Expression: /library/book

Name	XPath	Type
Title	title	String
Author	author/@name	String

	Title	Author
1	Everyday Italian	Giada De Laurentiis
2	Harry Potter	J K. Rowling
3	XQuery Kick Start	James McGovern
4	Learning XML	Erik T. Ray

4.2.2.3 Mapping for Nested XML Data

The can automatically detect some common types of nested XML Data as described in Section 3.8 Using the sample data in this section, if the user defines row mapping expression as:

/Order/OrderItem

Assuming that the sample data is available to the data set designer, the designer will be able to detect that the row element's parent element, "/Order", occurs in a repeating pattern. The designer therefore marks attributes and child elements of the "/Order" elements as candidates for columns.

5. ODA Extension Design

Runtime Driver plugin id & source package name

org.eclipse.birt.report.data.oda.xml

Designer plugin id & source package name

org.eclipse.birt.report.data.oda.xml.ui

XML Data Source Model Extension Properties

Name	Type	Comments
url	String	URL of XML data. Use "file" protocol for flat files
validating	Boolean	If true, runtime data is validated against the XML schema (provided by the schema property)
schema	String	Stores the content of the XML Schema provided by user at design-time. Used to validate runtime XML data if validating == true

XML Data Set Extension Properties

Name	Type	Comments
queryText	(intrinsic)	queryText defines mapping rules for the data set's table and its columns, in the form of an XML fragment like the following: <table expression="/library/book" > <column name="Title" expression="&quot;title&quot;" dataType="string"/> <column name="Price" expression="&quot;title&quot;" dataType="number"/> …

		</table>