Mapping XML Documents to Tables

DataDirect Connect® for ODBC XML Driver

Introduction

The DataDirect Connect® *for* ODBC XML driver uses one of two modes when mapping XML documents to tables. The default mode maps a document to a single table. This mode works well if the target document has a simple structure that maps to rows and columns. In this mode, the table name is equivalent to the document name.

Because many XML documents are not structured in such a simple form, the driver also has a mode that maps the document to one or more tables. In this mode, the DataDirect Connect *for* ODBC XML driver uses an algorithm to correlate the document to the tables, columns, and rows. This paper explains the fundamentals of the algorithm.

Document Mapping Procedures

The DataDirect Connect *for* ODBC XML driver uses the following series of procedures when mapping XML documents to tables.

Locations

When you configure the driver data source, a series of locations is recorded. Each location identifies a full path to an XML document. This path can be on the local filing system or it can be a URL to a Web server.

The types of locations are:

Folder	Implies that each XML file is a single table. When defining a Folder location, you specify only a directory as the location (not a directory and a file name), for example, C:\xmlsample.
XML Document	Implies that the full path to the XML document, including the XML file name, is the location. Using this type of location, each document can have one or more tables and can be a hierarchical-formatted XML document. When defining an XML Document location, you specify a path and an XML file name, for example, C:\xmlsample\file.xml, as the location.

DataDirect TECHNOLOGIES

HTML Document	Implies the use of an HTML document with embedded XML data islands. Using this type of location, each document can have one or more tables. When defining an HTML Document location, you specify a path and an HTML file name, for example, C:\htmlsample\file.html, as the location.

Table Guessing Scan

When the driver connects, all documents referenced in the data source are accessed, and the driver loads each document into an MSXML DOM object. The driver uses the DOM object to scan each document and locate the root elements in the document for each potential table. This scan is referred to as the Table Guessing Scan. The result of the Table Guessing Scan is a list of paths to elements in the document that are each considered root elements for a table, as well as a name for each table.

Starting at the root of the document, the driver scans the entire document using a depth-first search. The starting point for this scan can be overridden using the Table Hint option on the Advanced tab of the Configure Location dialog box (Top Table Hint connection string attribute). For a limited set of documents, the starting point is known; for example, for documents generated by ADO persistence, the scan begins at the “data:rs” element.

The scan progresses from the root out to all of the children. As the scan begins, each element is considered a possible candidate for a table root element. For each element, the scan generates a list of all the distinct child elements. For each distinct child element, a count of the number of times the element name appears is kept with each distinct element name.

Using this list of distinct child elements, the scan tabulates a count of the number of repeating and non-repeating child elements for the current element. At this point, the scan determines whether or not the current element should be a root for a table.

If the current element is the root element for the document, and the number of repeating child elements is zero, the scan adds the document to the list of table roots. If the element has repeated child elements and non-repeated child elements, then the scan considers the current element a root element for a table. In addition, if there are any distinct child elements that are repeating, then those distinct child elements are added as additional tables to the list of table roots.

The scan proceeds to the next element in the depth-first search of the entire document, following the same analysis pattern. Once complete, the result is a list of XPATH paths to each element in the document that the scan considers to be the root element of a table. The scan keeps a name for the table with each path. The name is usually taken from the root element of the table.

The following is an example of an XML document:

```
<?xml version="1.0"?>
  <purchaseOrder orderDate="2003-10-20">
    <shipTo country="US">
      <name>Alice Smith</name>
      <street>123 Maple Street</street>
      <city>Mill Valley</city>
      <state>CA</state>
      <zip>90952</zip>
    </shipTo>
    <billTo country="US">
       <name>Robert Smith</name>
       <street>8 Oak Avenue</street>
       <city>Old Town</city>
       <state>PA</state>
       <zip>95819</zip>
    </billTo>
    <comment>Hurry, my lawn is going wild!</comment>
    <items>
       <item partNum="872-AA">
          <productName>Lawnmower</productName>
          <quantity>1</quantity>
          <USPrice>148.95</USPrice>
          <comment>Confirm this is electric</comment>
       </item>
       <item partNum="926-AA">
          <productName>Baby Monitor</productName>
          <quantity>1</quantity>
          <USPrice>39.98</USPrice>
          <shipDate>2003-05-21</shipDate>
       </item>
    </items>
  </purchaseOrder>
```

The XML driver returns two tables: “purchaseOrder” and “items.” Two tables are returned because two items are found for a single purchase order. The XML driver found commonality of child elements.

Columns Scan

When the ODBC application accesses one of the tables in the table list, the driver assembles a list of columns. The most common access scenario is a SELECT statement on the table.

Using the XPATH associated with the table, the driver obtains from the DOM a list of child elements to the table. Each element in this list is considered the root to a record for the table. The scan generates a column list by inspecting each record. The Max Rows to Scan option on the Advanced tab of the Configure Location dialog box (Scan Rows connection string attribute) controls the number of records inspected for this scan. The more sample column values the scan encounters, the more accurate the determination. By default, the value is 0, which means the driver scans all rows in the table.

Limiting the number of rows can reduce the amount of time it takes to determine the column information on very large documents; however, because less information is available, the determination may be less accurate.

Before the actual columns are added to the list of columns, two manufactured columns are added to the table. First, the driver adds a primary key column called "_ID." This column is omitted if only one table is discovered in the document. Second, the driver adds a foreign key column if the table is a child table to a parent table. This column name is generated by concatenating the parent table name to the "_ID." The driver considers the table to be the parent to a child table if the table's root element is an ancestor to the child root element.

The driver performs a "find columns" scan for each record element node: first, the element is inspected for attribute columns; second, the element is inspected for child element nodes. The scan continues recursively until it either reaches a leaf node or encounters a child table. The column names are manufactured using the attribute name or the node element name. If the item is taken from more than one level below the root of the record, the parent's element name is prefixed to the column name.

A collection of columns is assembled in this manner as each record is scanned. If new columns are discovered as more and more records are encountered, the driver assumes that the previous records have NULL values for that column.

Consider the previous example document. The items table will receive two generated columns, `_ID` and `_purchaseOrder_ID`, which are assigned an integer data type. The purchaseOrder table receives only the `_ID` column, because it does not have a parent table.

The tables returned from the example file include the following columns:

Table	Columns	
items	_ID	quantity
	_purchaseOrder_ID	USPrice
	partNum	comment
	productName	shipDate
purchaseOrder	_ID	billTo_country
	orderDate	billTo_name
	shipTo_country	billTo_street
	shipTo_name	billTo_city
	shipTo_street	billTo_state
	shipTo_city	billTo_zip
	shipTo_state	comment
	shipTo_zip	

Data Type Guessing

As the columns scan encounters each column for each row, a guess must be made about the data type of the column. Using the value of the column, a determination of the data type is made. A type-guessing function evaluates each value for the column. The type returned by the guessing function is compared to the current best guess for the column's type, which is taken from previous observed values. The lowest common denominator of all types is string. The final best guess is the type used when the scan is complete. For example, if the column value always has numeric digits, the guessing code guesses that the value is an integer.

The data type guessing function limits its data types to a subset of the DataDirect Format data types, as listed in the following table.

Data Type	Sample Values
wvchar	"Foo", "best320"
varbinary	"27AB2F9C"
int	"34","-7000"
unsignedint	"0", "123456789"
long	"-12345678012345"
unsignedlong	"123456789012345"
boolean	"true", "false"
date	1963-12-19
time	10:09:58
timeinstant	1963-12-19T10:09:58
decimal	1245.678

Summary

Many XML documents have a complex structure that cannot be mapped to a single table. To accommodate these documents, the DataDirect Connect *for* ODBC XML driver uses a mode in which the document is mapped to one or more tables. In this mode, the driver uses a table-guessing algorithm to find the tables, columns, and rows in the document.

We welcome your feedback! Please send any comments concerning documentation, including suggestions for other topics that you would like to see, to:

docgroup@datadirect.com

FOR MORE INFORMATION

800-876-3101

info@datadirect.com

Worldwide Sales

Belgium (French).....0800 12 045
Belgium (Dutch).....0800 12 046
France.....0800 911 454
Germany 0800 181 78 76
Japan 0120.20.9613
Netherlands.....0800 022 0524
United Kingdom.....0800 169 19 07
United States.....800 876 3101

DataDirect Technologies is focused on data access, enabling software developers at both packaged software vendors and in corporate IT departments to create better applications faster. DataDirect Technologies offers the most comprehensive, proven line of data connectivity components available anywhere. Developers worldwide depend on DataDirect Technologies to connect their applications to an unparalleled range of data sources using standards-based interfaces such as ODBC, JDBC and ADO.NET, as well as cutting-edge XML query technologies. More than 250 leading independent software vendors and thousands of enterprises rely on DataDirect Technologies to simplify and streamline data connectivity. DataDirect Technologies is an operating company of Progress Software Corporation (Nasdaq: PRGS).

www.datadirect.com