Super-Fast XML Wrapper Generation in DB2: A Demonstration

Vanja Josifovski, Sabine Massmann, and Felix Naumann
IBM Almaden Research Center, San Jose, CA 95120
vanja@almaden.ibm.com, smassma@us.ibm.com, felix@almaden.ibm.com

Abstract

The XML Wrapper is a new feature of the federated database capabilities of DB2/UDB v8. It enables users and applications to issue SQL queries against XML data from a variety of sources, including files and web services. The XML Wrapper assumes hierarchical XML documents modeled as families of virtual relational tables in a federated schema, which can then be queried to extract information from the XML and combine it with data from other sources.

Due to the nature of the problem, using the XML Wrapper is complex and several difficult steps must be undertaken: (i) The hierarchical schema of the source must be flattened to a relational form. (ii) Each relation of the flattened schema must be registered in DB2 as a NICKNAME *– a complex virtual table definition containing several XPaths as specialized options. (iii) Each* NICKNAME *must be accompanied by a* VIEW *– again a complex structure involving join conditions. Chocolate is a tool that alleviates all three tasks: Chocolate provides several flattening strategies and an interface allowing users to modify the automatically generated target schema. Once the user is satisfied with the schema, Chocolate automatically generates the corresponding* NICKNAME *and* VIEW *definitions.*

1 The XML Wrapper

In response to the dramatically increasing production and consumption of XML data IBM offers the XML Wrapper, which uses the federated query processing capabilities of IBM's DB2/UDB database [3]. The XML Wrapper enables optimized execution of SQL queries across an integrated schema that can include both relational data and XML data from a variety of sources [2].

The core idea behind the XML Wrapper is that a hierarchical XML document can be thought of as a set of relational tables in which hierarchical nesting is replaced by joins between parent and child tables. The XML data is not stored in DB2, but is retrieved and processed on demand from its source, e.g., by reading an XML document from a file or by sending a SOAP request over the network.

When the query is executed, the wrapper retrieves a stream of XML data from the source and extracts the information requested by the query. This data is returned to the middleware engine in a relational form, where it can be filtered, aggregated, joined with other data, etc.

NICKNAMES. The XML data is mapped to a set of virtual relational tables, called NICKNAMES, in the integrated schema. The XML Wrapper uses XPath [1] expressions to establish a correspondence between portions of the XML document and rows in the tables, and to identify values within those document portions that correspond to each row's columns. The SQL statements of Figure 1 are example NICKNAME definitions for customer elements and order elements, originally nested under customers. This hierarchy is expressed as the implicit key-foreign-key relationship between the two NICKNAMES.

To write such NICKNAME definitions the author must be proficient in writing SQL statements, be aware of the structure of the XML document to decide which elements and attributes should be part of the customer NICKNAME. Furthermore, the author must be familiar with XPath expressions which are used for each attribute and for the NICKNAME itself. Finally the author must be knowledgeable of the XML type-system and the SQL type-system to decide which types the NICKNAME attributes should have. The Chocolate tool performs all these tasks automatically.

VIEWS. Queries against NICKNAMES of the XML Wrapper must include the complete join path between each of the NICKNAMES queried and the root NICKNAMEof the hierarchy. For instance, a query against `order_nickname` alone would be invalid. Instead, the query would have to include the join condition `order_nickname.customer_FID = customer_nickname.customer_id`. To alleviate this additional burden, a VIEW is defined over each NICKNAME, and DB2's query rewrite mechanism is exploited to transform queries submitted against these views to the required format. Below is a VIEW definition for

```
 CREATE NICKNAME customer_nickname
  ( name        VARCHAR(48)  OPTIONS(XPATH './name/string()'),
    address     VARCHAR(48)  OPTIONS(XPATH './@address'),
    customer_id VARCHAR(16)  OPTIONS(PRIMARY_KEY 'YES') )
  FOR SERVER xml_server
  OPTIONS(XPATH     '//customer',
          FILE_PATH 'C:\Chocolate\customer.xml');

CREATE NICKNAME order_nickname
 ( amount       DOUBLE      OPTIONS(XPATH './amount/text()'),
   date         VARCHAR(48) OPTIONS(XPATH './date/text()'),
   order_ID     VARCHAR(48) OPTIONS(PRIMARY_KEY 'YES'),
   customer_FID VARCHAR(48) OPTIONS(FOREIGN_KEY 'customer_nickname'))
 FOR SERVER xml_server
 OPTIONS(XPATH './/order');
```

Figure 1. Two NICKNAME **statements**

`order_nickname`. Note the join condition that establishes the connection between `order_nickname` and `customer_nickname`.

```
CREATE VIEW order AS
    SELECT order.amount, order.date,
           order.order_ID,
           customer.customer_ID
    FROM   order_nickname order,
           customer_nickname customer
    WHERE  customer.customer_ID
            = order.customer_FID;
```

Again, the task of writing such VIEW definitions is quite complex. The author must be familiar with SQL view definitions, and additionally with the hierarchy of the NICKNAMES, which reflects the hierarchy of the XML document. Again, Chocolate automatically generates the appropriate VIEWS.

An additional obstacle in the rapid deployment of the XML Wrapper is often the sheer number of necessary definitions. A common XML schema in the life sciences domain is the SwissProt XML Schema [4]. Converting each repeating element of this schema into a NICKNAME and a VIEW results in 98 definitions and some 70 kByte of SQL code. The Chocolate tool completely automates all these steps within seconds, while retaining the ability of the user to intervene when needed.

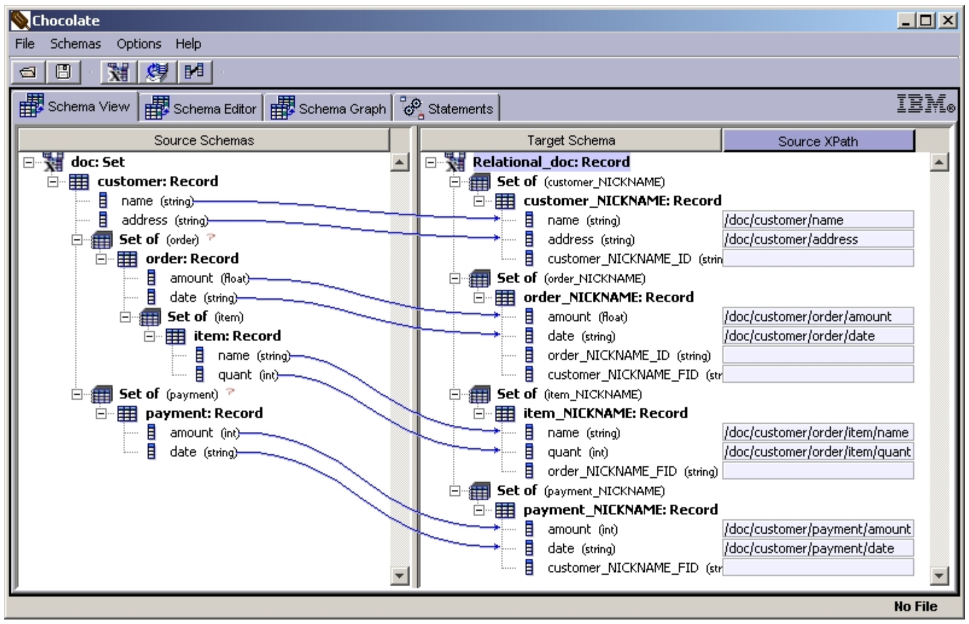
2 Mapping XML Schemas to the Relational Model

To map an XML document to a set of tables, its schema must be mapped to a set of (virtual relations). This mapping should be lossless in the data, i.e., all data of the document has a place in the tables. This mapping should also be lossless in the structure, i.e., no association implied by the original hierarchy shall be lost. In addition to the definition of the mapping, the names of the tables and their attributes must be decided. There must also be a translation from the XML data types to the relation data types of DB2. Chocolate performs all these tasks in two steps. First, the entire XML schema is mapped to a set of virtual relations. Next, the user can edit this schema through various operations like adding and deleting attributes, changing types, etc.

Automatic Strategies. Chocolate supports two strategies of automatically generating a relational schema from an XML schema. The first strategy is called *full normalization* and creates a relation for every repeating set. Repeating sets are identified in the XML schema either through the `<sequence>` tag or through the `maxoccurs` attribute of an XML element. Figure 2 shows a screenshot of the Chocolate GUI with the result of this strategy applied to an XML schema. The left half of the GUI shows the original XML schema as a tree. A repeating set is shown as a `Set` node. The direct children of each `Set` node form the attributes of the corresponding relation. The right half of GUI shows the resulting relational schema as a set of NICKNAMES. In addition to the relational schema, we see for each regular attribute the XPaths showing the source of the attribute data. Additional attributes are the virtual keys and foreign keys, used by the XML Wrapper to reconstruct the hierarchy of the XML document.

The second strategy to form a relational schema is called *single relation*. Here, only a single relation is formed with an attribute for each leaf element of the XML schema. In this case, no virtual keys or foreign keys are necessary. In general, there are many more potential automatic strategies, ranging anywhere between *single table* and *full normalization* and even beyond. Currently we are researching a strategy that optimizes response time of queries through the XML Wrapper.

User Modifications. After having automatically generated a relational schema, Chocolate allows the user to mod-



```
Source Schemas                         Target Schema                          Source XPath
doc: Set                               Relational_doc: Record
  customer: Record                       Set of (customer_NICKNAME)
    name (string)                          customer_NICKNAME: Record
    address (string)                         name (string)                    /doc/customer/name
    Set of (order)                           address (string)                 /doc/customer/address
      order: Record                          customer_NICKNAME_ID (strin
        amount (float)                   Set of (order_NICKNAME)
        date (string)                      order_NICKNAME: Record
        Set of (item)                        amount (float)                   /doc/customer/order/amount
          item: Record                       date (string)                    /doc/customer/order/date
            name (string)                    order_NICKNAME_ID (string)
            quant (int)                      customer_NICKNAME_FID (str
    Set of (payment)                     Set of (item_NICKNAME)
      payment: Record                      item_NICKNAME: Record
        amount (int)                         name (string)                    /doc/customer/order/item/name
        date (string)                        quant (int)                      /doc/customer/order/item/quant
                                             order_NICKNAME_FID (string)
                                         Set of (payment_NICKNAME)
                                           payment_NICKNAME: Record
                                             amount (int)                     /doc/customer/payment/amount
                                             date (string)                    /doc/customer/payment/date
                                             customer_NICKNAME_FID (str
```

Figure 2. Chocolate – Automatic mapping from hierarchical schema to relational schema

ify this target schema in a variety of ways. Available modification operations are, Add/Delete attribute, Add/Delete NICKNAME, Rename attribute/NICKNAME, Change data type, and Split/merge NICKNAMES. Not all operations are valid at all times. Chocolate checks their feasibility and either warns the user or prohibits the operation. Once the user is satisfied, the modified schema can be saved and reloaded at a later time.

Users have additional options to influence the NICKNAME statement generation, such as, changing the schema name, the default length for data of the XML string type, etc.

Mapping Web-Services to the Relational Model. The XML Wrapper can process queries against XML data from many different sources: XML documents can be stored locally in a file system, or in a database column, or they can be retrieved remotely using a Web-Service. The former two choices are expressed as special single-line options in the NICKNAME statement. The latter is somewhat more complex, because Web-Services (i) are defined by a different type of schema (WSDL), (ii) are activated by sending complex messages, (iii) and expect input parameters in the XML format. Chocolate alleviates these difficulties by being able to parse the WSDL, generate the correct NICKNAME options containing the Web-Service SOAP calls etc., and by defining a SQL user-defined function that takes input parameters and tags them appropriately as XML input for the Web-Service.

3 The Chocolate Demo

The demonstration of Chocolate will comprise the loading and conversion of many different schemas, including small schemas to demonstrate special cases, and very large schemas like the SwissProt XML schema [4] to demonstrate scalability. We will showcase the different strategies of mapping an XML schema to the relational model and let users modify the converted schema. The automatically generated NICKNAME and VIEW statements will be registered in DB2 and queries will be executed against these virtual tables. In particular the usage of large schemas and large XML documents will demonstrate the advantage of Chocolate to rapidly deploy an new XML data source in DB2, and will demonstrate the usefulness ad efficiency of the XML Wrapper in issuing SQL queries against XML documents.

References

[1] J. Clark and S. DeRose. *XML Path Language (XPath) Version 1.0 – W3C Recommendation*. World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[2] V. Josifovski and P. Schwarz. Querying XML data sources in DB2: The XML Wrapper. 2003. Submitted to ICDE 2003 industrial track.

[3] V. Josifovski, P. Schwarz, L. Haas, and E. Lin. Garlic: A new flavour of federated query processing in DB2. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, Madison, WN, 2002.

[4] SP-ML: The SWISS-PROT/TrEMBL XML Format. `http://www.ebi.ac.uk/swissprot/SP-ML/`.